МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
имени М.В. ЛОМОНОСОВА
ФАКУЛЬТЕТ ВЫЧИСЛИТЕЛЬНОЙ МАТЕМАТИКИ И КИБЕРНЕТИКИ

На правах рукописи

Глонина Алевтина Борисовна

АНАЛИЗ КОНФИГУРАЦИЙ МОДУЛЬНЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ ДЛЯ ПРОВЕРКИ ВЫПОЛНЕНИЯ ОГРАНИЧЕНИЙ РЕАЛЬНОГО ВРЕМЕНИ

Специальность 05.13.11 —
Математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей

ДИССЕРТАЦИЯ
на соискание ученой степени
кандидата физико-математических наук

Научный руководитель:
кандидат физико-математических наук
Балашов Василий Викторович

Москва — 2020

Оглавление

Стр.

Стр.

Введение

Актуальность работы

В настоящее время перспективной архитектурой информационно-управляющих систем (ИУС) реального времени является модульная архитектура [1]. Если для более старой федеративной архитектуры характерно использование для каждого приложения собственного специализированного вычислителя, то в системах с модульной архитектурой аппаратные ресурсы разделены между приложениями. Вычислители унифицированы и функционируют под управлением операционных систем со стандартизованным программным интерфейсом. По сравнению с федеративной архитектурой, модульная архитектура позволяет более эффективно использовать аппаратные ресурсы, сократить массу, размеры, энергопотребление системы, производить разработку приложений независимо от аппаратной реализации вычислителей.

Модульные вычислительные системы (МВС) применяются для управления летательными аппаратами, автомобилями, поездами, медицинскими аппаратами и иными сложными техническими системами [2, 3]. Известны работы по созданию перспективных МВС для использования в различных областях промышленности [4—6]. В данной работе МВС рассматриваются на примере систем интегрированной модульной авионики (ИМА) [2, 7].

В данной работе модульные вычислительные системы (МВС) рассматриваются на примере систем ИМА. Рабочая нагрузка на систему ИМА обычно представлена набором задач, сгруппированных в разделы. Под разделом понимается группа прикладных задач, как правило соответствующих одному приложению, и взаимодействующих посредством общей памяти. Задачи выполняются периодически: один раз за период должен выполниться экземпляр задачи, называемый работой. Для МВС должны выполняться *ограничения реального времени*: все работы должны успеть выполниться в рамках заданных директивных интервалов, определяемых периодами задач. Информационный обмен между работами разных задач осуществляется посредством передачи сообщений. Между задачами, имеющими одинаковый период, могут существовать зависимости по данным: очередная работа задачи-получателя может быть поставлена на выполнение в рамках своего директивного интервала только после получения сообщений от всех соответствующих работ задач-отправителей.

Доступ разделов к ресурсам вычислителя осуществляется согласно статическому расписанию окон выполнения разделов. Окно — интервал времени, в течение которого могут выполняться работы одного заданного раздела. Расписание окон строится при проектировании МВС, с помощью внешних по отношению к МВС инструментальных средств. В рамках окон раздела выполнением работ управляет сопоставленный разделу динамический планировщик.

Под конфигурацией МВС понимается: количество и типы вычислителей; набор разделов, в т.ч. характеристики входящих в них прикладных задач (приоритеты, периоды и т.п.) и зависимости по данным между ними; привязка разделов к вычислителям; расписание окон разделов.

В данной работе рассматривается задача проверки выполнения ограничений реального времени для МВС с заданной конфигурацией. Одним из широко используемых на практике методов проверки выполнения таких ограничений является построение и анализ временно́й диаграммы

(ВД) функционирования этой МВС при длительностях выполнения работ прикладных задач и обменов данными, равных заданным верхним оценкам. ВД содержит интервалы выполнения работ. Получив такую ВД, можно непосредственно проверить соответствие интервалов выполнения работ ограничениям реального времени. Именно этот метод используется в данной работе.

ВД функционирования МВС с заданной конфигурацией может быть построена посредством прогона имитационной модели функционирования такой МВС.

В современных летательных аппаратах на смену федеративной архитектуре ИУС пришла модульная архитектура. При проектировании МВС возникает задача проверки выполнения ограничений реального времени для конфигураций таких систем. В частности, такая задача возникает при инкрементальной разработке программного обеспечения МВС — требуется проверить, что при изменении рабочей нагрузки (добавлены новые задачи, поменялись длительности выполнения задач и/или передачи сообщений) можно использовать ранее построенное расписание окон и не требуется формировать его заново.

Степень разработанности темы работы

Решению задач, возникающих при проектировании ИУС реального времени, посвящен ряд работ отечественных и зарубежных исследователей. В область моделирования, верификации и выбора конфигураций таких систем (в т.ч. решения задач планирования) значительный вклад внесли работы Н. П. Бусленко, В. М. Глушкова, В. Е. Котова, В. В. Воеводина, Р. Л. Смелянского, Э. Г. Коффмана, Б. Г. Сушкова, М. Г. Фуругяна, Н. В. Колесова, В. А. Костенко, Ю. Г. Карпова, В. А. Захарова, А. К. Петренко, И. А. Ломазовой, R. Milner, C. A. R. Hoare, J. Bergstra, T. Henzinger, R. Alur и D. Dill, P. Dissaux и F. Singhoff, E. André, K. Larsen, E. Clarke, C. L. Liu и J. Layland, N. Audsley и A. Burns, и др. Разработка МВС для конкретных бортов выполняется коллективами следующих организаций: Корпорация «Иркут», Компания «Сухой», ГосНИИАС, НПП «Полет», Airbus, Boeing и др. Операционные системы для МВС разрабатываются коллективами ГосНИИАС совместно с ИСП РАН, Компании «Сухой», ИТМиВТ совместно с «СВД—Встраиваемые системы», НИИСИ РАН, РПКБ, ОКБ «Электроавтоматика».

Существует несколько подходов к решению задачи проверки выполнения ограничений реального времени для МВС с заданной конфигурацией.

Известен ряд методов, основанных на методе анализа времени отклика (RTA — Response Time Analysis) [8—10]. Такие методы называются аналитическими. Несколько аналитических методов было разработано для МВС [11—14]. Однако, все перечисленные методы не учитывают того, что в разных разделах могут использоваться различные алгоритмы планирования. В то же время данная возможность реализована в таких операционных системах для систем ИМА, как VxWorks 653 [15] и Багет 3 [16]. Кроме того, в работах [11, 13, 14] не рассматриваются зависимости по данным между прикладными задачами, а в работе [12] рассматриваются зависимости по данным лишь между задачами одного раздела.

Другим подходом к проверке выполнения ограничений реального времени для МВС с заданной конфигурацией является построение модели системы и ее формальная верификация. Этот подход позволяет учесть все аспекты функционирования МВС, однако он не применим для систем большой размерности, так как его вычислительная сложность экспоненциально зависит от

количества прикладных задач в системе. Так в работе [17], где рассматривается упрощенный вариант задачи проверки ограничений реального времени (к каждому вычислителю привязан только один раздел), для конфигурации с 20 задачами процесс верификации занимает уже 2,5 минуты, с 25 задачами — 5 минут. Таким образом, процесс верификации конфигурации с 100 задачами займет более полутора лет, в то время как реальные системы содержат несколько сотен задач. Результаты экспериментов, выполненных автором настоящей работы для задачи проверки выполнения ограничений реального времени в общем случае (без ограничений на количество разделов, привязанных к вычислителю) и приведенные в [18], также подтвердили неприменимость данного подхода для систем большой размерности: время верификации модели МВС, включающей 18 задач, на персональном компьютере составляет более трех минут и увеличивается в два раза с добавлением в конфигурацию одной задачи. Таким образом, верификация модели МВС, включающей 40 задач, займет уже более двух лет.

Еще одним методом проверки выполнения ограничений реального времени для МВС с заданной конфигурацией является построение ВД функционирования этой МВС при длительностях выполнения прикладных задач и обменов данными, равных заданным верхним оценкам. ВД должна содержать интервалы выполнения работ прикладных задач. Получив такую ВД, можно непосредственно проверить выполнение ограничений реального времени. Этот метод проверки ограничений реального времени широко применяется при решении ряда задач проектирования МВС (например, [19—22]). В данной работе рассматривается именно этот метод, поскольку он, в отличие от методов, основанных на формальной верификации моделей, и от аналитических методов, одновременно удовлетворяет таким важным требованиям как пригодность для анализа конфигураций большой размерности и учет всех существенных для проверки ограничений реального времени особенностей организации МВС (в т.ч. наличие зависимостей по данным между задачами и использование разных планировщиков в разных разделах).

ВД функционирования МВС с заданной конфигурацией может быть построена посредством прогона имитационной модели функционирования такой МВС. При построении этой модели ее корректность (то есть соответствие поведения модели поведению целевой МВС) должна быть доказана математически, так как на этапе проектирования МВС отсутствует возможность экспериментального сравнения поведения модели и поведения целевой системы. Согласно современным принципам разработки программного обеспечения, модель должна иметь модульную структуру (по аналогии с целевой системой), то есть формироваться из моделей типовых компонентов МВС с возможностью включения в свой состав пользовательских моделей компонентов МВС. В результате анализа представленных в открытом доступе методов и средств моделирования вычислительных систем [21, 23—26] не было найдено средства, в полной мере удовлетворяющего всем сформулированным требованиям. Поэтому было принято решение о разработке новых методов и средств решения поставленной задачи.

Цели и задачи

Целью диссертационной работы является разработка методов и средств проверки выполнения ограничений реального времени для МВС с заданной конфигурацией.

Для достижения поставленной цели были выделены следующие задачи:

– Провести обзор задач, возникающих при проектировании МВС и требующих проверки выполнения ограничений реального времени для конфигураций МВС; сформулировать требования к методам и средствам такой проверки, а также математическому аппарату в их основе.
– Выбрать математический аппарат для описания функционирования МВС и расширить его в соответствии со сформулированными требованиями.
– Построить обобщенную модель функционирования МВС, абстрагированную от структуры МВС и используемых в разделах алгоритмов планирования.
– Доказать корректность построенной обобщенной модели.
– Предложить метод, конкретизирующий обобщенную модель для МВС с заданной конфигурацией и позволяющий при помощи полученной модели проверить выполнение ограничений реального времени для этой конфигурации МВС (с учетом конкретных алгоритмов планирования, используемых в разделах).
– Разработать инструментальную систему, реализующую предложенный метод, и выполнить экспериментальное исследование разработанного метода на данных, соответствующих реальным МВС.

Объектом исследования являются МВС и их конфигурации. **Предметом исследования** — методы проверки выполнения ограничений реального времени для МВС с заданной конфигурацией.

Научная новизна и теоретическая ценность

В данной работе используемый при моделировании функционирования МВС математический аппарат сетей временных автоматов с остановкой таймеров расширен для абстрагирования от систем переходов автоматов. Это позволило разработать обобщенную модель функционирования МВС, абстрагированную от структуры МВС и используемых в МВС алгоритмов планирования, и доказать ее корректность. Разработан метод проверки выполнения ограничений реального времени для заданной конфигурации МВС, основанный на конкретизации обобщенной модели МВС для заданной конфигурации МВС. Предложен и успешно применен подход к доказательству корректности обобщенной модели функционирования МВС. Результаты данной работы могут быть применены в качестве базы для построения и проверки корректности моделей функционирования вычислительных систем других классов (например, сочетающих элементы федеративной и модульной архитектур).

Практическая ценность

Практическая ценность работы обусловлена тем, что разработана инструментальная система с открытым исходным кодом[1], позволяющая получать ВД функционирования МВС с заданной конфигурацией и проверять выполнение ограничений реального времени для этой конфигурации МВС. Инструментальная система может быть использована как совместно с системами автоматизированного проектирования МВС, так и самостоятельно.

Методы исследования

При получении основных результатов диссертации использовались методы теории автоматов, формальной верификации моделей (Model Checking), математической статистики.

[1] https://github.com/AlevtinaGlonina/MCSSim

Положения, выносимые на защиту

1. Обобщенная модель функционирования МВС, абстрагированная от структуры МВС и используемых в МВС алгоритмов планирования. Модель базируется на аппарате сетей временных автоматов с остановкой таймеров, расширенном в работе для абстрагирования от систем переходов автоматов. Доказана корректность обобщенной модели функционирования МВС.
2. Метод проверки выполнения ограничений реального времени для заданной конфигурации МВС. Метод конкретизирует обобщенную модель функционирования МВС для заданной конфигурации и использует полученную модель при построении временной диаграммы функционирования МВС.
3. Инструментальная система проверки выполнения ограничений реального времени для конфигураций МВС, разработанная на основе предложенного метода. Экспериментальное исследование подтвердило применимость разработанного метода для анализа конфигураций МВС реальной размерности.

Апробация работы

Результаты, представленные в работе, докладывались на научных семинарах кафедры Автоматизации систем вычислительных комплексов факультета ВМК МГУ под руководством чл.-корр. РАН профессора Р.Л. Смелянского, а также на 8 конференциях:

1. Международной конференции «Ломоносов 2015»;
2. Всероссийской конференции «Ломоносовские чтения 2016»;
3. Международной конференции «ORM 2016»;
4. Всероссийской конференции «Ломоносовские чтения 2017»;
5. Международной конференции «РАСТ 2017»;
6. Международной конференции «Суперкомпьютерные дни в России 2017»;
7. Всероссийской конференции «Ломоносовские чтения 2018»;
8. Всероссийской конференции «Тихоновские чтения 2019».

Степень достоверности результатов

Достоверность представленных результатов обеспечивается математическими доказательствами корректности формируемых моделей. Также для всех проанализированных в экспериментальном исследовании конфигураций МВС временные диаграммы, полученные с помощью этих моделей, совпали с временными диаграммами, полученными с помощью встроенной в САПР «Планировщик ИМА» [27] модели.

Публикации

Основные результаты по теме диссертации изложены в 14 печатных изданиях [28—41]. 5 публикаций [28—32] изданы в журналах, рекомендованных ВАК, 3 из них [28, 29, 32] изданы в журналах, индексируемых Scopus / Web of Science / RSCI. Инструментальная система проверки выполнения ограничений реального времени для конфигураций модульных вычислительных систем была зарегистрирована в установленном законодательством порядке [42].

Личный вклад автора

Подготовка части материалов к публикации проводилась совместно с соавторами, причем вклад диссертанта был определяющим.

Вклад соавторов публикаций заключается в следующем. В работе [29] вклад Балашова В. В. заключается в постановке задачи обоснования корректности разработанных моделей. В работах [32, 38] Бахмурову А. Г. принадлежит схема формального представления конфигурации МВС. В работе [35] вклад Балашова В. В. заключается в описании архитектуры систем ИМА, Бахмурова А. Г. — в редактуре раздела, посвященного моделированию функционирования вычислительных модулей. В работе [40] Балашову В. В. принадлежит описание структуры ИУС реального времени с архитектурой ИМА.

Все представленные в диссертации результаты получены лично автором.

Структура работы

Диссертация состоит из введения, пяти глав, заключения и четырех приложений.

Во введении обоснована актуальность диссертационной работы, дано краткое описание задачи, сформулирована цель и аргументирована научная новизна исследований, показана практическая значимость полученных результатов, представлены выносимые на защиту научные положения.

В первой главе описаны устройство и примеры МВС, на основе обзора и анализа задач, возникающих при проектировании МВС и требующих проверки выполнения ограничений реального времени для конфигураций МВС, сформулированы требования к инструментальным средствам имитационного моделирования МВС. Также в первой главе введены формальные определения конфигурации МВС и временной диаграммы МВС с заданной конфигурацией, сформулировано условие выполнения ограничений реального времени. Приведена формальная постановка задачи проверки выполнения ограничений реального времени для МВС с заданной конфигурацией.

Во второй главе сформулированы требования к математическому аппарату для моделирования МВС, проведен обзор математических аппаратов, по результатам обзора выбран аппарат временных автоматов с остановкой таймеров. Предложены: обобщенная модель функционирования МВС, абстрагированная от структуры МВС и используемых в МВС алгоритмов планирования; расширение математического аппарата сетей временных автоматов с остановкой таймеров для абстрагирования от систем переходов автоматов (обобщенные сети временных автоматов с остановкой таймеров), позволившее построить обобщенную модель и доказать ее корректность; метод проверки выполнения ограничений реального времени для конфигурации МВС, конкретизирующий обобщенную модель для заданной конфигурации МВС и использующий построенную модель при получении ВД функционирования МВС. Приведены асимптотические оценки сложности построения и прогона моделей.

В третьей главе предложен подход к доказательству корректности обобщенной модели функционирования МВС и доказана корректность этой модели. На основе доказанных в данной главе утверждений сделан вывод о том, что для всех построенных согласно предложенному подходу моделей МВС с заданной конфигурацией получаемые ВД соответствуют ВД функционирования целевой МВС, и, следовательно, такие модели могут использоваться для проверки выполнения ограничений реального времени.

В четвертой главе описана разработанная автором инструментальная система проверки выполнения ограничений реального времени для МВС, в которой реализован предложенный подход

к проверке выполнения ограничений реального времени для конфигураций МВС. Описана схема интеграции этой системы с САПР «Планировщик ИМА».

В пятой главе описана методика экспериментального исследования предложенных методов и средств, приведены результаты экспериментального исследования.

В заключении сформулированы основные результаты работы и указаны направления развития исследований.

В приложении А подробно описаны разработанные автором модели компонентов МВС.

В приложении Б сформулированы и доказаны утверждения о выборе значений параметров и переменных, использующиеся при верификации моделей компонентов МВС.

В приложении В описаны разработанные автоматы-наблюдатели, использующиеся для проверки выполнения требований корректности к моделям компонентов МВС.

В приложении Г представлены результаты экспериментов в табличном виде.

Полный объем диссертации составляет 245 страниц (127 страниц не считая приложений), включая 62 рисунка и 8 таблиц. Список литературы содержит 141 наименование.

Глава 1. Постановка задачи проверки выполнения ограничений реального времени для МВС

1.1. Устройство и примеры МВС

Опишем в данном разделе устройство модульных вычислительных систем (МВС) реального времени на примере систем интегрированной модульной авионики (ИМА).

МВС, в том числе системы ИМА, состоят из стандартизированных модулей, каждый из которых содержит один или несколько процессоров, как правило многоядерных. Процессоры могут различаться типами и количеством ядер, а ядра разных типов могут различаться производительностью. Взаимодействие между модулями осуществляется посредством коммутируемых сетей с виртуальными каналами (например [43, 44]).

Рабочая нагрузка на процессорные ядра представлена набором прикладных задач. Все задачи являются периодическими: за период должен выполниться один экземпляр задачи, называемый работой. Задачи объединены в разделы — группы логически связанных прикладных задач, взаимодействующих посредством общей памяти. Как правило, один раздел соответствует одному приложению. Каждый раздел привязан к вычислительному ядру в составе процессора. Несколько разделов могут быть привязаны к одному и тому же ядру. Для того, чтобы исключить взаимное влияние разделов, привязанных к одному ядру, в системах ИМА реализовано пространственное и временное разделение аппаратных ресурсов. Пространственное разделение заключается в том, что каждому разделу соответствует своя область памяти, недоступная другим разделам. Взаимодействие между задачами разных разделов осуществляется только посредством передачи сообщений. Временное разделение заключается в том, что на интервале планирования для вычислительного ядра задается статическое расписание окон, то есть отрезков времени, в течении каждого из которых могут выполняться работы определенного раздела. Интервал планирования, на котором строится расписание окон, равен, как правило, наименьшему общему кратному периодов всех задач МВС. По истечении интервала планирования расписание окон повторяется. На рисунке 1.1 показаны вычислительные модули МВС с привязанной к ним рабочей нагрузкой.

Каждый раздел имеет собственный планировщик, управляющий постановкой на выполнение работ внутри окон раздела. Как правило, в системах ИМА используются динамические планировщики. Так стандарт ARINC 653 [45] предполагает использование динамического планировщика с фиксированными приоритетами и вытеснением. Известны системы, в которых применяются планировщики, работающие по стратегии EDF с вытеснением [46], а также планировщики с фиксированным приоритетами без вытеснения [47]. Допускается использование разных алгоритмов планирования в разных разделах. Для каждой задачи и каждого типа ядра известно наихудшее (максимальное) время выполнения одиночной работы данной задачи на ядре данного типа. В действительности время выполнения работ может быть меньше этого времени,

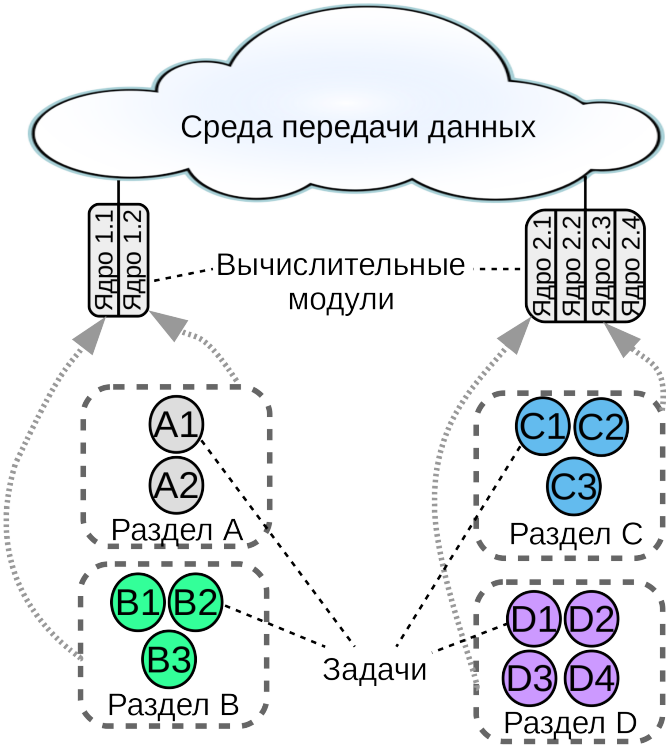

Рисунок 1.1 — Вычислительные модули МВС с привязанной к ним рабочей нагрузкой

однако на практике [19, 48] при планировании вычислений принято считать, что для каждой работы длительность выполнения на ядре фиксирована и равна заданной величине. У каждой работы существует директивный интервал, определяющийся периодом задачи, номером работы, а также смещениями левой и правой границ этого интервала относительно начала периода. При этом для задачи, имеющей период p, директивный интервал i-й работы ($i \geqslant 1$) принадлежит отрезку $[(k-1) \cdot p, k \cdot p]$. Работа становится готовой к выполнению не ранее определенной для нее левой границы директивного интервала. Работа должна завершиться не позднее правой границы своего директивного интервала. Если правая граница директивного интервала достигается до завершения выполнения работы, то работа считается *опоздавшей* и не может далее выполняться на вычислительном ядре. В таком случае, время, которое опоздавшая работа находилась на вычислительном ядре, меньше заданного времени, необходимого для ее выполнения.

Между задачами с одинаковым периодом могут существовать синхронные зависимости по данным: очередная работа задачи-получателя считается не готовой к выполнению до тех пор, пока не получит данные от всех соответствующих работ задач-отправителей, а также пока текущее время не достигнет левой границы директивного интервала этой работы. Данные передаются в сообщениях фиксированного (для каждой пары задач) размера. Для каждого сообщения известны верхние оценки длительности передачи через внутреннюю память модуля и через сеть. По аналогии с длительностями выполнения работ, при планировании вычислений принято считать, что для каждого сообщения длительность передачи фиксирована и равна одной из двух (в зависимости от того, привязаны ли отправитель и получатель к одному модулю) указанных оценок. Для коммутрируемых сетей с виртуальными каналами существуют достаточно точные методы получения таких оценок. Так в работе [49] описан метод оценки длительности передачи сообщений

AFDX, позволивший на реальных данных получить верхние оценки, отличающиеся от точных значений не более, чем на 12%.

Кроме того, между задачами могут существовать асинхронные зависимости по данным: работы задач-отправителей посылают сообщения работам задач-получателей, однако работам задач-получателей не требуются «свежие» данные для начала выполнения; работа задачи-получателя, все входные зависимости которой являются асинхронными, становится готовой, как только текущее время достигает левой границы директивного интервала этой работы. Асинхронные зависимости могут существовать между задачами с разными периодами. Наличие таких зависимостей по данным не влияет на порядок выполнения и, следовательно, время завершения работ, поэтому далее такие зависимости рассматриваться не будут. При этом влияние асинхронных сообщений на длительность передачи синхронных сообщений считается учтенным в верхних оценках длительностей передачи синхронных сообщений. Также в верхних оценках длительностей передачи синхронных сообщений учтено влияние топологии сети на эти длительности, поэтому далее топология сети отдельно рассматриваться не будет.

Рисунок 1.2 иллюстрирует схему планирования вычислений в системах ИМА. На верхней части рисунка представлена рабочая нагрузка: разделы A (задачи $A1$, $A2$), B (задачи $B1$, $B2$) и C (задачи $C1$, $C2$, $C3$). Числовая часть имени задачи соответствует ее приоритету: $A1$ — приоритет равен 1, $C3$ — приоритет равен 3 и т.д. Чем больше числовое значение приоритета задачи, тем выше ее приоритет среди задач раздела. Периоды всех задач одинаковы. Стрелками показаны синхронные зависимости по данным: задача $C2$ зависит по данным от задачи $C1$, $B2$ — от задачи $C3$. Разделы A и B привязаны к первому ядру, раздел C — ко второму. Постановкой на выполнение работ каждого раздела управляет планировщик с фиксированными приоритетами и вытеснением. На нижней части рисунка представлены окна разделов для каждого вычислительного ядра и порядок выполнения работ внутри окон. Окна выделены цветами, соответствующими разделам.

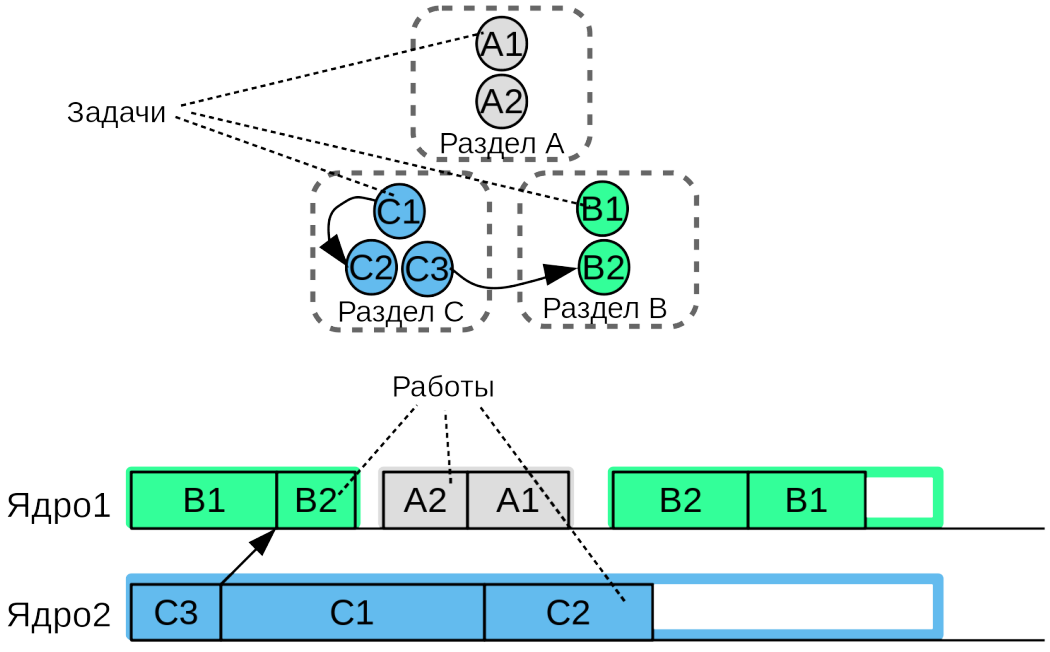

Рисунок 1.2 — Схема планирования в системах ИМА

Приведем несколько примеров систем ИМА, разрабатываемых и используемых в отечественной промышленности.

Известен ряд работ по разработке аппаратной составляющей систем ИМА. Так ФГУП «ГосНИИАС» разработал [50] семейство систем ИМА, аппаратная часть которых включает 16 вычислительных модулей и 4 модуля коммутации AFDX [43], а ОАО «Авиационная холдинговая компания «Сухой» — семейство систем интегрированной модульной авионики боевых комплексов [51], каждая из которых содержит 6 или 8 вычислительных модулей (2 или 4 универсальных вычислительных процессорных модуля, 2 модуля графического контроллера и 2 модуля ввода/-вывода) и 2 или 4 модуля коммутации. Для использования в этом семействе систем была создана бортовая цифровая вычислительная машина на основе принципов интегрированной модульной авионики боевых комплексов для ИУС самолета Су-57 [52].

Согласно предложениям ФГУП «ГосНИИАС» [53] комплекс бортового оборудования (КБО) гражданского самолета на базе ИМА имеет структуру, представленную на рисунке 1.3. Архитектуру ИМА в данном случае имеет большинство вычислительных систем КБО: вычислительное ядро КБО, система индикаторов, интегрированная система наблюдения, интегрированный комплекс связи, интегрированный радионавигационный комплекс.

На 4-й Международной конференции «Перспективные направления развития бортового оборудования гражданских воздушных судов» коллективом ФГУП «ГосНИИАС» была представлена перспективная структура КБО на базе распределенной модульной авионики (РМА) [54]. Согласно данному предложению, КБО включает четыре системы ИМА: вычислительное ядро КБО, радиосистема ИМА CNS, интегрированная система ИМА интеллектуальной поддержки экипажа, интегрированная система ИМА управления техническим состоянием летательного аппарата.

Модульный бортовой комплекс средств цифровой радиосвязи с архитектурой ИМА предложен [55] НПП «Полет».

ПАО «Корпорация «Иркут» разработала [56] интегрированный комплекс бортового оборудования разнородной архитектуры, в составе которого имеется вычислительное ядро, имеющее архитектуру ИМА и состоящее из шести вычислительных модулей и двух модулей коммутации, и ряд систем, имеющих федеративную архитектуру.

В ПАО «ОАК — центр комплексирования» совместно с ПАО «Корпорация «Иркут» была создана интегрированная вычислительная система самолета МС-21 [57], содержащая 6 вычислительных модулей и 4 коммутатора. Рабочая нагрузка представлена 7 приложениями, соответствующими различным разделам.

Согласно [2] разработкой МВС с архитектурой ИМА также занимаются такие предприятия как АО «Раменское приборостроительное конструкторское бюро», ЗАО НТЦ «Модуль», ОАО «НИИ ВК им. М.А.Карцева» и др.

Примерами отечественных операционных систем, поддерживающими архитектуру ИМА, являются Багет 3 [16], БагрОС-4000 [58], JetOS [59], ЭОС [60].

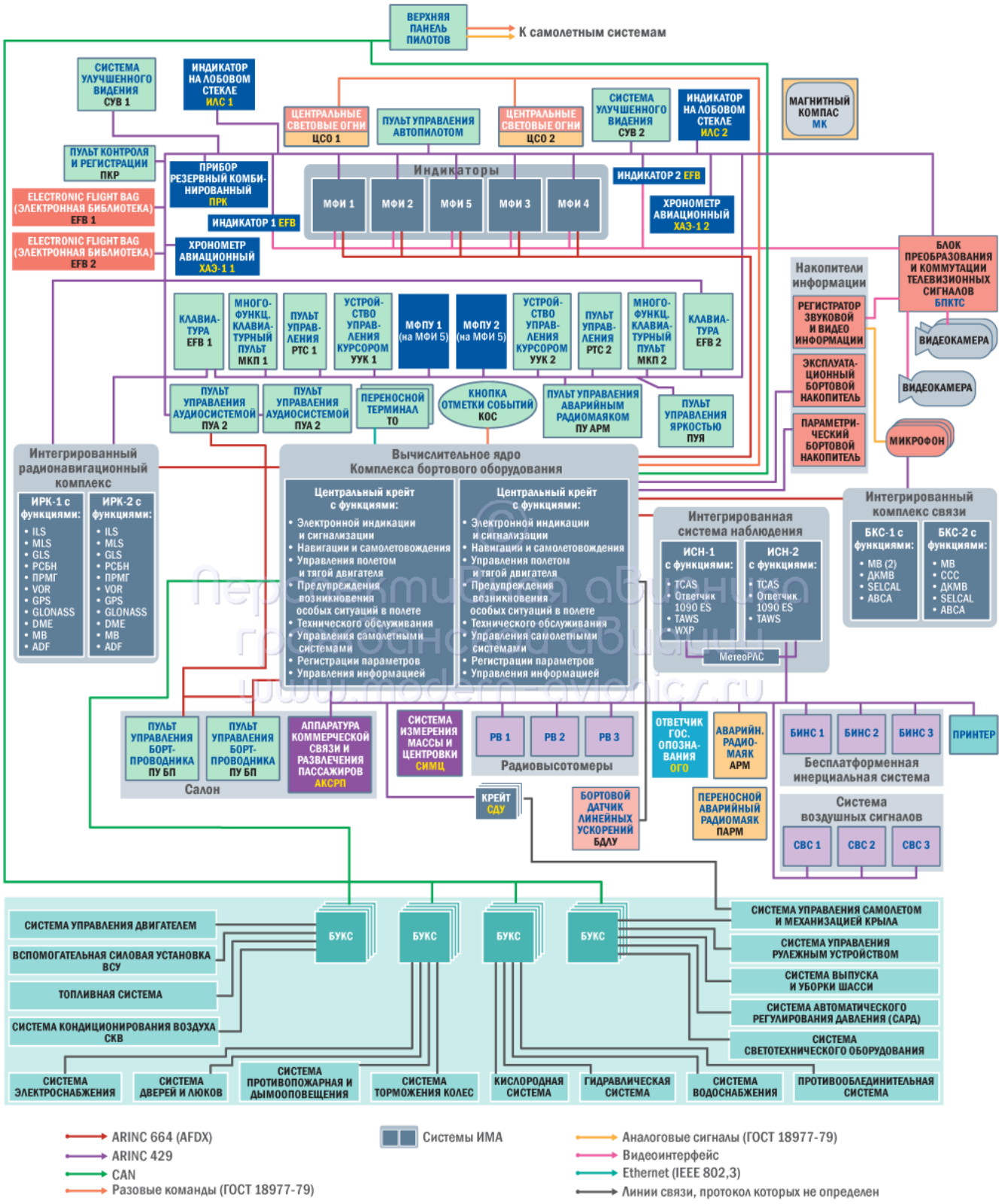

Рисунок 1.3 — Структура КБО на базе ИМА согласно [53]

1.2. Конфигурация МВС и ограничения реального времени

Введем в данном разделе содержательное определение конфигурации МВС на примере систем ИМА. Определения конфигураций для других МВС, примеры которых перечислены во введении, могут незначительно отличаться от представленного здесь. В данной работе в определение конфигурации МВС включены лишь те характеристики МВС, которые существенны для задачи проверки выполнения ограничений реального времени.

Конфигурация МВС определяет состав и характеристики процессорных ресурсов с их разделением по модулям, состав и характеристики рабочей нагрузки, ее распределение по ядрам, а также расписания окон для каждого ядра.

Для каждого модуля в конфигурации указаны типы *процессорных ядер* и количество ядер каждого типа. Тип ядра определяет его производительность.

Рабочая нагрузка в конфигурации представлена набором задач, сгруппированных в разделы, и набором синхронных сообщений. Согласно рассуждениям, приведенным выше в разделе 1.1, асинхронные сообщения в данной работе не рассматриваются.

Для каждого раздела указан алгоритм планирования работ, использующийся в этом разделе. Напомним, что как правило [45—47] для планирования работ внутри раздела используются известные алгоритмы динамического планирования, такие как планирование на основе фиксированных приоритетов с вытеснением [8, 45] и без вытеснения [61], планирование на основе директивных сроков (EDF) [8]. Для каждого из этих алгоритмов планирования существует формальное описание, например в виде псевдокода [62] или конечного автомата [45, 63]. Будем считать, что алгоритм динамического планирования работ внутри раздела задан в виде такого формального описания.

Для каждой задачи в конфигурации указаны следующие характеристики:

- уникальный в рамках раздела приоритет;
- для каждого типа ядра — наихудшая (максимальная) длительность выполнения на ядре этого типа;
- смещение относительно начала периода, определяющее левые границы директивных интервалов работ задачи;
- смещение относительно начала периода, определяющее правые границы директивных интервалов работ задачи; данное смещение не превышает длительность периода.

Для некоторых систем, например, рассматриваемых в работе [19], у большинства задач первое смещение равно 0, а второе — периоду задачи. В этом случае длительность директивных интервалов равна периоду задачи.

Для каждого синхронного сообщения в конфигурации указаны следующие характеристики:

- задача-отправитель;
- задача-получатель;
- наихудшая (максимальная) длительность передачи сообщения через память модуля;
- наихудшая (максимальная) длительность передачи сообщения через сеть.

Напомним, что в данной работе асинхронные сообщения, а также топология сети не рассматриваются как отдельные составляющие конфигурации, так как их влияние уже учтено при оценке наихудших длительностей передачи синхронных сообщений.

Для каждого раздела в конфигурации указано ядро, к которому привязан раздел (то есть задана *привязка разделов к ядрам*). Также для каждого раздела задан набор окон на интервале планирования, в течение которых могут выполняться работы задач данного раздела (то есть задано *расписание окон разделов*).

Для конфигурации МВС *выполняются ограничения реального времени*, если для нее в процессе функционирования МВС ни одна из работ не становится опоздавшей.

Как было отмечено во введении, проверку выполнения ограничений реального времени для некоторой конфигурации целесообразно выполнять посредством анализа *временной диаграммы* функционирования МВС с данной конфигурацией (ВД МВС). ВД определена на интервале планирования и представляет собой набор событий, наблюдаемых в системе. Каждое событие характеризуется типом, источником (то есть компонентом МВС, к которому относится событие) и временем. В данной работе рассматриваются события, длительность которых не существенна, поэтому каждому событию сопоставляется одно значение времени. Для проверки выполнения ограничений реального времени представляют интерес события следующих типов: постановка работы на выполнение, вытеснение и завершение работы. Работа может иметь несколько интервалов выполнения. Каждый *интервал выполнения работы* представляет собой либо интервал между постановкой работы на выполнение и ее вытеснением, либо интервал между постановкой работы на выполнение и ее завершением (под постановкой работы на выполнение понимается в том числе и возврат вытесненной работы на вычислительное ядро).

Неформально условие выполнения ограничений реального времени для некоторой конфигурации МВС может быть сформулировано следующим образом: для конфигурации выполняются ограничения реального времени, если для каждой работы суммарная длительность ее интервалов выполнения равна максимальной длительности ее выполнения на ядре, к которому привязан соответствующий раздел. Нарушение данного условия заключается в том, что некоторые работы являются опоздавшими. Для проверки выполнения этого условия достаточно построить и проанализировать ВД МВС. При этом анализ ВД МВС для каждой работы заключается в получении суммарной длительности интервалов выполнения и сравнении этой длительности с заданной в конфигурации величиной, а основная задача заключается в построении ВД МВС.

Таким образом, возникает задача построения ВД МВС для заданной конфигурации МВС в условиях недоступности самой МВС. Искомая ВД может быть построена посредством прогона имитационной модели МВС, которую также необходимо построить.

1.3. Актуальность задачи

На различных этапах проектирования могут варьироваться различные составляющие конфигурации МВС, вследствие чего возникает большое число потенциальных конфигураций МВС.

На начальных этапах проектирования возникает задача выбора количества и типов модулей. Для фиксированного набора модулей необходимо выбрать наилучшее распределение разделов по ядрам. Далее для каждого ядра строится расписание окон разделов. Кроме того, требуется назначить приоритеты прикладным задачам. При решении каждой из перечисленных задач проектирования, как правило, происходит анализ большого числа возможных конфигураций МВС, для которых необходимо проверять выполнение ограничений реального времени.

В общем случае такие задачи проектирования имеют вид задачи выбора наилучшей по некоторому критерию конфигурации, удовлетворяющей ограничениям реального времени (например, в [19, 48, 64—69]). Такого рода задачи решаются с использованием специальных программных средств (САПР), имеющих в основе оптимизационные алгоритмы. Примерами минимизируемых критериев оптимизации являются нагрузка на среду передачи данных [19, 64], число переключений контекста [48, 69], стоимость [66, 68], масса [66, 68] и энергопотребление [66] системы, число модулей [68].

В цикле работы оптимизационных алгоритмов необходимо осуществлять проверку выполнения ограничений реального времени для потенциальных конфигураций МВС.

Кроме того, при проектировании МВС рассматриваются задачи поиска конфигурации, удовлетворяющей ограничениям реального времени, среди множества потенциальных конфигураций без выделения оптимизируемого критерия (например, такая задача решается в [20, 70, 71]). При решении таких задач также требуется проверять выполнение ограничений реального времени для потенциальных конфигураций.

Также задача проверки выполнения ограничений реального времени возникает при инкрементальной разработке программного обеспечения МВС — требуется проверить, что при изменении рабочей нагрузки (добавлены новые задачи, поменялись длительности выполнения задач и/или передачи сообщений) можно использовать ранее построенное расписание окон и не требуется формировать его заново. Поддержка такого подхода реализована, например, в средстве [27].

В перечисленных работах варьируется число и типы модулей [68], распределение разделов по ядрам [19, 20, 64, 65, 67, 68, 71], расписания окон разделов [19, 48, 64, 67, 69, 71], назначение приоритетов [19, 70].

Автором был проанализирован ряд задач ([19, 20, 48, 64—71]), возникающих при проектировании МВС и требующих проверки выполнения ограничений реального времени для конфигураций. В результате данного анализа было выявлено, что в большинстве перечисленных работ при проверке ограничений не учитываются такие аспекты функционирования, как синхронные зависимости задач по данным, а в некоторых — даже логика планирования работ внутри разделов. В таких предположениях для проверки выполнения ограничений реального времени могут быть применены аналитические методы, которые и используются в указанных работах. При таком подходе, на дальнейших этапах проектирования системы, когда будут учтены перечисленные аспекты функционирования системы, а также при испытаниях и эксплуатации системы, может оказаться, что выбранная ранее конфигурация не удовлетворяет ограничениям. Лишь в [19] и [20] указанные выше аспекты функционирования МВС учитываются. В этих работах, а также в [69] ограничения реального времени для конфигураций проверяются посредством анализа

временных диаграмм, получаемых в результате прогона имитационных моделей МВС. Однако подходы к построению моделей, применяемые в [19, 20, 69], не позволяют строго обосновать корректность моделей и, следовательно, корректность проверки ограничений реального времени. Кроме того, средства, описанные в [19, 20, 69], не имеют открытого кода, а следовательно непригодны для исследования и модификации.

По результатам анализа перечисленных задач проектирования и работ, в которых они решаются, автором были сформулированы следующие требования к инструментальным средствам имитационного моделирования МВС:

1. *Возможность получения ВД, включающих события постановки на выполнение, вытеснения и завершения работ.* Такой ВД достаточно для проверки выполнения ограничений реального времени для конфигурации МВС, сформулированного в разделе 1.2.
2. *Математический аппарат в основе средства должен позволять описывать те аспекты функционирования МВС, которые существенны для проверки выполнения ограничений реального времени*. В частности необходима возможность моделирования вытеснения работ, синхронных зависимостей по данным, использования различных алгоритмов планирования в различных разделах (поддержка различных алгоритмов планирования в различных разделах реализована, например, в операционных системах VxWorks653 [15], Багет 3 [16]), использования нескольких типов вычислительных ядер, различающихся производительностью.
3. *Математический аппарат в основе средства должен предоставлять возможность проверки корректности моделей математическими методами*. Такая проверка необходима, так как на этапе проектирования МВС невозможно экспериментально сравнить поведение модели с поведением реальной МВС.
4. *Поддержка библиотеки моделей компонентов МВС*. Различные МВС строятся из стандартизированных компонентов, поэтому и модели МВС целесообразно строить из моделей этих компонентов, образующих библиотеку моделей.
5. *Автоматизация построения и прогона модели.* Для того, чтобы система моделирования могла использоваться в цикле работы оптимизационных алгоритмов, процесс построения модели по описанию конфигурации, а также прогона модели и анализа результатов моделирования должен быть полностью автоматизирован.
6. *Возможность включения в модель МВС и в библиотеку моделей пользовательских моделей компонентов МВС.* В процессе проектирования может потребоваться проанализировать конфигурации, включающие компоненты (планировщики, среды передачи данных и т.п.), моделей которых нет в библиотеке моделей. Для этого у проектировщика должна быть возможность самостоятельно создать новые модели компонентов системы.
7. *Скорость* построения и прогона модели, а также анализа результатов моделирования должна позволять работать с конфигурациями размерности, соответствующей реальным системам, с учетом числа конфигураций, оцениваемых при проектировании (это число может достигать нескольких сотен и более при работе оптимизационных алгоритмов).

Автором был проведен обзор ряда средств моделирования МВС, в результате которого в открытом доступе не было выявлено ни одного средства, в полной мере удовлетворяющего

перечисленным требованиям. Проанализированы следующие средства, имеющиеся в открытом доступе: Cheddar [23], HSSim [24, 72], MAST [25], MASIW [26], ДИАНА [73] и ДИАНА-2012 [21].

Cheddar [23] не поддерживает моделирование снятия работы по завершению ее директивного интервала, использование пользовательских моделей задач и компонентов сетей передачи данных (поддерживаются только пользовательские модели планировщиков), формальную проверку корректности моделей. Также Cheddar имеет только графический интерфейс. Таким образом, это средство не удовлетворяет требованиям 2, 3, 5 и 6.

HSSim [24, 72] не поддерживает моделирование синхронных зависимостей по данным и не позволяет задать статическое расписание окон, не поддерживает формальную проверку корректности моделей. Таким образом, это средство не удовлетворяет требованиям 2 и 3.

MAST [25] поддерживает ограниченный набор алгоритмов планирования, не позволяет задать статическое расписание окон, не поддерживает формальную проверку корректности моделей, использование пользовательских моделей компонентов МВС и имеет только графический интерфейс. Таким образом, это средство не удовлетворяет требованиям 2, 3, 5 и 6.

MASIW [26] поддерживает структурный анализ моделей — формальную проверку требований к структуре моделей (требования описываются на языке REAL), но не верификацию поведения моделей (важность такой верификации обоснована в главе 3; там же приведен ряд требований к поведению моделей). Таким образом, требование 3 для этого средства выполняется лишь частично. Кроме того, средство MASIW реализовано как набор расширений для графической среды Eclipse, что затрудняет автоматизацию построения и прогона моделей (требование 5).

ДИАНА [73] и ДИАНА-2012 [21] не поддерживают моделирование вытеснения и, следовательно, не удовлетворяют требованию 2.

Анализ перечисленных средств показал, что их доработка для поддержки всех перечисленных требований значительно более трудоемка, чем создание нового средства.

Также были рассмотрены частично или полностью закрытые средства, описанные в публикациях [19, 20, 74, 75]. Они не удовлетворяют требованию 3, т.е. не позволяют формально проверять корректность моделей.

Таким образом, анализ предметной области показал, что актуальна задача разработки методов и средств проверки выполнения ограничений реального времени для МВС с заданной конфигурацией, удовлетворяющих всем перечисленным требованиям. Отметим, что данные требования относятся как к математическим моделям, лежащим в основе методов и средств, так и к их программным реализациям. Так требования 1—3 относятся в большей степени к математическим моделям, а 4—7 — к программным реализациям.

1.4. Формальная постановка задачи

1.4.1. Определение конфигурации МВС

Для того, чтобы формализовать задачу проверки выполнения ограничений реального времени для МВС с заданной конфигурацией, необходимо ввести строгое определение конфигурации МВС. Неформальное определение конфигурации МВС было дано в разделе 1.2.

Введенное ниже формальное определение конфигурации МВС основано на модели распределенной вычислительной системы, предложенной в работах Р. Л. Смелянского [22].

Обозначим за $\mathbb{N}_0$ множество натуральных чисел ($\mathbb{N}$), расширенное числом 0.

Пусть $N_m \in \mathbb{N}$ — количество модулей в рассматриваемой МВС.

Пусть $HW = \{HW_i\}_{i=1}^{N}$ — *набор вычислительных ядер*, $HW_i = \langle type_i, mod_i \rangle$, где $N \in \mathbb{N}$ — количество вычислительных ядер в МВС, $type_i \in \overline{1,N_t}$ — тип i-го ядра, от которого зависит его производительность ($N_t \in \mathbb{N}$ — количество типов ядер), $mod_i \in \overline{1,N_m}$ — номер вычислительного модуля, которому принадлежит процессор, содержащий i-е ядро.

Именем $CType$ обозначим отображение, ставящее в соответствие некоторому ядру его тип:

$$CType : HW \to \overline{1,N_t}$$

Отметим, что для рассматриваемой задачи тип модуля характеризуется количеством и типами процессорных ядер, поэтому отдельно типы модулей в формальное определение конфигурации не включены.

Введем обозначения для *рабочей нагрузки*, которая состоит из набора задач, сгруппированных в разделы, и информации о зависимостях по данным между задачами. Рабочая нагрузка будет представлена в виде кортежа $WL = \langle Part, Msg \rangle$, где $Part$ — набор разделов, Msg — набор синхронных сообщений, передаваемых между задачами.

Пусть $A = \{A_j\}_{j=1}^{N_a}$ — набор динамических алгоритмов планирования, которые могут использоваться для планирования работ разделов ($N_a \in \mathbb{N}$ — количество алгоритмов планирования). Каждый алгоритм из A имеет формальное описание логики его работы.

Набор разделов $Part$ представляется в виде $Part = \{Part_i\}_{i=1}^{M}$, $Part_i = \langle T_i, a_i \rangle$, где

- $M \in \mathbb{N}$ — количество разделов;
- $T_i = \{T_{ij}\}_{j=1}^{K_i}$ — набор задач i-го раздела, $T_{ij} = \langle pr_{ij}, \overline{c_{ij}}, p_{ij}, o_{ij}, d_{ij} \rangle$ — атрибуты j-й задачи i-го раздела, где:
  - $K_i \in \mathbb{N}$ — количество задач в i-ом разделе;
  - $pr_{ij} \in \mathbb{N}$ — уникальный в рамках раздела приоритет задачи ($\forall i \in \overline{1,M}$ $\forall j_1,j_2 \in \overline{1,K_i}, j_1 \neq j_2 : pr_{ij_1} \neq pr_{ij_2}$);
  - $\overline{c_{ij}} = (c_{ij}^1, ..., c_{ij}^{N_t}) \in \mathbb{N}^{N_t}$ — длительности выполнения задачи в худшем случае (WCET) на разных типах вычислительных ядер;
  - $p_{ij} \in \mathbb{N}$ — период задачи;

- $o_{ij} \in \mathbb{N}_0, o_{ij} < p_{ij}$ — смещение относительно начала периода, определяющее левые границы директивных интервалов работ задачи: для k-й работы задачи T_{ij} левая граница директивного интервала равна $p_{ij} * (k-1) + o_{ij}$;
- $d_{ij} \in \mathbb{N}, o_{ij} < d_{ij} \leqslant p_{ij}$ — смещение относительно начала периода, определяющее правые границы директивных интервалов работ задачи: для k-й работы задачи T_{ij} правая граница директивного интервала равна $p_{ij} * (k-1) + d_{ij}$;

– $a_i \in \overline{1, N_a}$ — задает алгоритм планирования, использующийся в разделе (а именно, a_i — номер алгоритма в наборе A).

Обозначим за $\hat{T}$ множество всех задач в МВС: $\hat{T} = \cup_{i=1}^{M} T_i$.

Набор сообщений Msg представляется в виде $Msg = \{Msg_j\}_{j=1}^{H}$, где H — количество синхронных сообщений, передаваемых между задачами. Сообщение Msg_j представляется в виде кортежа $Msg_j = \langle S_j, R_j, dm_j, dn_j \rangle$, где

– $S_j \in \hat{T}$ — задача-отправитель;

– $R_j \in \hat{T}$ — задача-получатель;

– $dm_j \in \mathbb{N}_0$ — длительность передачи сообщения через память модуля;

– $dn_j \in \mathbb{N}_0$ — длительность передачи сообщения через сеть.

Отображение $Bind : Part \to HW$ задает *привязку разделов к ядрам*. Каждый раздел привязан к одному ядру.

На интервале планирования длительностью L ($L \in \mathbb{N}$) задано *расписание окон разделов* в виде $Sched = \{\{\langle start_{ij}, stop_{ij} \rangle\}_{j=1}^{N_i^w}\}_{i=1}^{M}$, где:

– $N_i^w \in \mathbb{N}$ — количество окон для i-го раздела;

– $start_{ij}, stop_{ij} \in \overline{0,L}$ — времена начала и завершения j-го окна i-го раздела.

L как правило равно наименьшему общему кратному всех p_{ij}. В связи с этим далее будем считать, что L кратно p_{ij} для любых $i \in \overline{1,M}\, \forall j \in \overline{1,N_i^w}$.

На расписание окон накладываются следующие ограничения корректности:

– Время начала каждого окна меньше времени его завершения: $\forall i \in \overline{1,M}\, \forall j \in \overline{1,N_i^w}$ $start_{ij} < stop_{ij}$.

– Окна различных разделов, привязанных к одному вычислительному ядру, не пересекаются, однако время завершения одного раздела может совпадать с временем начала другого. То есть $\forall i_1, i_2 \in \overline{1,M}$, $i_1 \neq i_2$, если $Bind(Part_{i_1}) = Bind(Part_{i_2})$, то $\forall j_1 \in \overline{1,N_{i_1}^w}$, $\forall j_2 \in \overline{1,N_{i_2}^w}$: либо $stop_{i_1 j_1} \leqslant start_{i_2 j_2}$, либо $stop_{i_2 j_2} \leqslant start_{i_1 j_1}$.

Во введенных выше обозначениях *конфигурация МВС* — это кортеж $\langle HW, WL, Bind, Sched \rangle$, где:

– HW — набор вычислительных ядер;

– WL — описание рабочей нагрузки;

– $Bind$ — привязка разделов к ядрам;

– $Sched$ — расписание окон разделов.

В данной работе предполагается, что время дискретно: все времена и длительности, являющиеся атрибутами элементов конфигурации, задаются в квантах времени. На практике квант

времени можно выбрать равным наибольшему общему делителю тактов процессоров, либо просто установить достаточно малым (например, равным 1 мкс) и округлить все длительности вверх до целого числа квантов времени.

1.4.2. Определение временной диаграммы МВС с заданной конфигурацией

Пусть $conf$ — некоторая конфигурация МВС. Определим формально ВД МВС с конфигурацией $conf$.

Для каждой задачи T_{ij} на интервале планирования длительности L определен набор работ $W_{ij} = \{w_{ijk}\}_{k=1}^{L/p_{ij}}$. Директивный интервал работы w_{ijk} равен $[(k-1)\cdot p_{ij} + o_{ij}; (k-1)\cdot p_{ij} + d_{ij}]$.

Введем определение события в МВС. Данное определение привязано к рабочей нагрузке, но едино для всех конфигураций МВС, в которые входит эта рабочая нагрузка. *Событием в МВС* будем называть кортеж $e = \langle EType, Src, t\rangle$, где:

- $EType \in \{EX, PR, FIN\}$ — тип события: постановка работы на выполнение (EX); вытеснение работы с ядра (PR); завершение работы, связанное с полным выполнением работы, либо с наступлением правой границы директивного интервала (FIN);
- $Src \in \bigcup\limits_i \bigcup\limits_j W_{ij}$ — работа-источник события;
- $t \in \mathbb{N}_0 \cap [0,L]$ — время события.

$E = \{EX, PR, FIN\} \times \bigcup\limits_i \bigcup\limits_j W_{ij} \times (\mathbb{N}_0 \cap [0,L])$ — множество всевозможных событий при выполнении заданной рабочей нагрузки.

На практике [19, 48] при проектировании МВС предполагают, что время выполнения каждой работы на соответствующем ядре фиксировано и равно наихудшему (максимальному возможному), время передачи каждого сообщения также фиксировано и равно наихудшему, а все используемые алгоритмы планирования детерминированы и результаты их работы не зависят от факторов, не описанных в конфигурации МВС. Примерами алгоритмов планирования, удовлетворяющих данному предположению, являются алгоритмы с фиксированными приоритетами (с вытеснением и без вытеснения) и алгоритмы, работающие по стратегии EDF (Earliest Deadline First). Стратегия EDF предполагает, что на выполнение ставится та работа из множества готовых работ, правая граница директивного интервала которой ближе к текущему моменту времени; при этом, если у нескольких готовых работ правые границы директивных интервалов совпадают, то выбор работы осуществляется однозначно, например, в соответствии с уникальными идентификаторами работ или порядком следования работ в очереди, который определяется детерминированно. Примерами алгоритмов планирования, не удовлетворяющих данному предположению, являются алгоритмы, осуществляющие выбор работы для постановки на выполнение недетерминированно (например, в соответствии с заданными вероятностями), а также алгоритмы, результаты работы которых зависят от времени получения сообщений от внешних подсистем. Алгоритмы, не удовлетворяющие сформулированному предположению, не рекомендуется использовать в вычислительных системах реального времени из-за неопределенности, вносимой в

выполнение работ самими алгоритмами, либо внешними подсистемами. В данной работе такие алгоритмы не рассматриваются. Введенные предположения соответствуют второму (системному) уровню детализации при проектировании вычислительных систем, введенному в работе [76].

Рассматриваемые в работе алгоритмы динамического планирования таковы, что данных из описания конфигурации МВС (атрибутов задач T_{ij} и сообщений Msg_j) достаточно для работы этих алгоритмов, с поправкой на формат представления.

Пусть $CONF$ — множество всевозможных конфигураций, имеющих вид, введенный в разделе 1.4.1.

В сформулированных выше предположениях набор алгоритмов планирования A *однозначно* определяет отображение $Q : CONF \to 2^E$. Однозначность этого отображения непосредственно следует из сформулированных предположений и широко применяется в работах по планированию вычислений (например, в [47, 77, 78]).

ВД для конфигурации $conf \in CONF$ — подмножество множества E, однозначно соответствующее $conf$, и являющееся результатом интерпретации A на $conf$.

Пусть $conf \in CONF$ — некоторая конфигурация системы, а $Q(conf)$ — ВД, соответствующая $conf$. В разделе 1.2 дано неформальное определение интервалов выполнения работы. Формально интервалами выполнения работы w_{ijk} считаются интервалы между событиями $\langle w_{ijk}, EX, t_0\rangle$ (постановка или возвращение работы на вычислительное ядро) и $\langle w_{ijk}, PR, t_1\rangle$ (вытеснение работы) и интервалы между событиями $\langle w_{ijk}, EX, t_0\rangle$ и $\langle w_{ijk}, FIN, t_1\rangle$ (завершение работы: штатное по окончанию вычислений, либо принудительное по достижению правой границы директивного интервала). Интервалы указанного вида не должны содержать внутри себя других событий типов EX, PR, FIN с источником w_{ijk}. Пусть R_{ijk} — количество интервалов выполнения работы w_{ijk}. Тогда упорядоченный набор событий для w_{ijk} имеет один из следующих видов:

- $\varnothing$, при $R_{ijk} = 0$ (работа не была поставлена на выполнение);
- $\{\langle w_{ijk}, EX, t_0\rangle, \langle w_{ijk}, FIN, t_1\rangle\} \subseteq Q(conf)$, при $R_{ijk} = 1$ (работа была поставлена на выполнение в момент t_0, ни разу не вытеснялась и завершилась в момент t_1);
- $\{\langle w_{ijk}, EX, t_0\rangle, \langle w_{ijk}, PR, t_1\rangle\} \subseteq Q(conf)$, при $R_{ijk} = 1$ (работа была поставлена на выполнение в момент t_0, была вытеснена в момент t_1 и больше не ставилась на выполнение, так как после вытеснения была достигнута правая граница ее директивного интервала);
- $\{\langle w_{ijk}, EX, t_0\rangle, \langle w_{ijk}, PR, t_1\rangle, ..., \langle w_{ijk}, EX, t_{2\cdot R_{ijk}-2}\rangle, \langle w_{ijk}, Y, t_{2\cdot R_{ijk}-1}\rangle\} \subseteq Q(conf)$, где $Y \in \{PR, FIN\}$, при $R_{ijk} > 1$ (работа была поставлена на выполнение в момент t_0, несколько раз вытеснялась, и завершилась, либо была в последний раз вытеснена в момент $t_{2\cdot R_{ijk}-1}$).

1.4.3. Условие выполнения ограничений реального времени для МВС

Условие выполнения ограничений реального времени для МВС с заданной конфигурацией заключается в том, что для каждой работы суммарная длительность ее интервалов выполнения

равна заданной максимальной длительности ее выполнения на вычислительном ядре соответствующего типа. Если условие выполняется, то работа завершилась штатно. Иначе работа была принудительно снята с ядра по достижению правой границы ее директивного интервала и является опоздавшей, то есть ограничения реального времени нарушены. В сформулированных в разделе 1.4.2 предположениях это условие является необходимым и достаточным.

В формальном виде условие имеет следующий вид:

$$RT(conf): \quad \forall w_{ijk}, i \in \overline{1,M}, j \in \overline{1,K_i}, k \in \overline{1,L/p_{ij}}: \\ R_{ijk} \geqslant 1 \text{ и } \sum_{r=1}^{R_{ijk}} (t_{2 \cdot r-1} - t_{2 \cdot r-2}) = c_{ij}^{CType(Bind(Part_i))}$$

1.4.4. Постановка задачи

Формально задача проверки выполнения ограничений реального времени для некоторой конфигурации МВС может быть поставлена следующим образом.

Дано: конфигурация МВС $conf \in CONF$.

Требуется: Определить, выполняется ли условие RT на конфигурации $conf$.

Согласно разделам 1.4.2, 1.4.3 для проверки условия $RT(conf)$ требуется построить ВД $Q(conf)$ функционирования МВС с конфигурацией $conf$. ВД $Q(conf)$ может быть построена посредством прогона имитационной модели МВС с заданной конфигурацией $conf$. В разделе 1.3 сформулированы требования, которым должны удовлетворять методы и средства моделирования МВС для того, чтобы их можно было использовать для проверки выполнения ограничений реального времени. В результате анализа описанных в литературе методов и средств было выявлено, что ни одно из них в полной мере не удовлетворяет данным требованиям. Поэтому актуальна задача разработки собственных методов и средств. Необходимые для решения этой задачи шаги изложены в разделе «Цель диссертационной работы» Введения.

Глава 2. Обобщенная модель функционирования МВС

При работе над данной главой диссертации использованы следующие публикации автора, в которых, согласно Положению о присуждении ученых степеней в МГУ, отражены основные результаты, положения и выводы исследования:

Glonina A., Bahmurov A. Stopwatch automata-based model for efficient schedulability analysis of modular computer systems // Parallel Computing Technologies (PaCT). Lecture Notes in Computer Science. — Springer, Cham, 2017. — Vol. 10421. — P. 289–300.

2.1. Выбор математического аппарата для построения модели

2.1.1. Требования к математическому аппарату

В разделе 1.3 сформулированы требования, которым должны удовлетворять инструментальные средства имитационного моделирования МВС. Требования 1—3 относятся к математическому аппарату, лежащему в основе инструментальных средств, а требования 4—7 — к программной реализации. Сформулируем на основе указанного списка требований требования к математическому аппарату для построения модели функционирования МВС:

1. *Возможность получения ВД, соответствующей модели и содержащей события с их временными метками*. Данное требование следует напрямую из требования 1 раздела 1.3.
2. *Возможность описания тех аспектов функционирования МВС, которые существенны для проверки выполнения ограничений реального времени*. Выбранный математический аппарат должен позволять описывать МВС на выбранном уровне абстракции, в том числе моделировать вытеснение работ, синхронные зависимости по данным, использование различных алгоритмов планирования в различных разделах, использование нескольких типов вычислительных ядер, различающихся производительностью (требование 2 раздела 1.3).
3. *Возможность проверки корректности моделей математическими методами* (требование 3 раздела 1.3).
4. *Наличие инструментального ПО для разработки, прогона и верификации моделей*. Создание такого ПО является трудоемкой задачей, которая не относится напрямую к задаче, решаемой в данной работе (проверка выполнения ограничений реального времени для конфигураций МВС). Поэтому целесообразно воспользоваться существующим ПО.

В следующих разделах (2.1.2—2.1.8) приведен обзор ряда математических аппаратов с целью выбора математического аппарата, в наибольшей степени удовлетворяющего перечисленным требованиям.

В разделе 1.4.2 сделан вывод о том, что на выбранном уровне абстракции ВД функционирования МВС с заданной конфигурацией определена однозначно. Поэтому в обзоре не будут рассматриваться математические аппараты, имеющие вероятностную природу, то есть предполагающих задание вероятности некоторых событий как количественной характеристики.

Функционирование МВС может быть представлено как функционирование дискретно-непрерывной системы. Дискретной составляющей при этом являются события, происходящие в системе (постановка на выполнение, вытеснение и завершение работ), а непрерывной составляющей — увеличение времени. Для моделирования МВС как дискретно-непрерывной системы могут быть использованы следующие классы математических аппаратов: гибридные автоматы [79], непрерывно-дискретные системы Глушкова [80], агрегативные системы Бусленко [81]. В работе [82] показано, как модели, построенные в одном из перечисленных аппаратов, могут быть представлены в любом из других двух аппаратов. Одним из недостатков этих аппаратов является то, что класс алгоритмически разрешимых для них задач верификации крайне беден [79]. Поэтому такие математические аппараты в общем виде не будут рассматриваться в данном обзоре. В обзор включены математические аппараты, позволяющие моделировать лишь частный случай дискретно-непрерывных систем, соответствующий МВС: единственной непрерывной составляющей таких систем является время, измеряемое с помощью специальных переменных-таймеров. В отличие от переменных общего вида, для таймеров определен лишь ограниченный набор операций, достаточный для моделирования МВС. В то же время эти ограничения позволяют решать задачи верификации для математических аппаратов с таймерами.

2.1.2. Временные автоматы и сети временных автоматов

Данный математический аппарат был предложен в работе [83] специально для моделирования функционирования вычислительных систем реального времени с целью анализа временных характеристик таких систем.

Аппарат временных автоматов является расширением аппарата конечных автоматов. Графически конечный автомат может быть представлен в виде размеченного ориентированного графа, вершины которого называют локациями, а дуги — переходами. В общем случае конечные автоматы позволяют моделировать последовательности состояний системы без учета длительности нахождения в этих состояниях. В [84] приведен обзор работ, в которых для моделирования времени используются обычные целочисленные переменные в рамках аппарата конечных автоматов. Этот подход обладает недостатками. Во-первых, при моделировании системы необходимо описывать отдельную модель времени, что усложняет процесс моделирования и приводит к более громоздким моделям целевых систем. Во-вторых, при неправильном выборе шага увеличения времени либо прогон и верификация модели могут иметь слишком большую вычислительную сложность (большинство состояний соответствует не различным «содержательным» состояниям системы, а состояниям, отличающимся лишь значениями времени), либо модель может быть

неточной (один временной шаг модели соответствует нескольким событиям, в реальной вычислительной системе происходящим в разное время).

Согласно [83], временной автомат представляет собой конечный автомат с заданным алфавитом, начальными и конечными локациями, а также таймерами. Таймер — это специальная переменная, принимающая вещественные значения. Для таймеров определены операции обнуления, сравнения с целочисленными значениями и другими таймерами. Начальное значение каждого таймера равно нулю. Значения таймеров увеличиваются согласованно на одну и ту же величину. Состояние автомата определяется его текущей локацией и значениями таймеров. Переходы размечены символами алфавита, условиями переходов являются ограничения на значения таймеров. При этом, если условие выполнено, то переход может, но не обязан быть выполнен. При выполнении перехода значение таймера может быть обнулено. Данные автоматы применяются для задания регулярных временных языков. Такие языки отличаются от обычных регулярных языков тем, что между символами слова временного языка присутствуют вещественнозначные задержки. В терминах вычислительных систем символами алфавита являются события, которые могут произойти в системе, задержками — длительности интервалов между событиями. Автомат задает язык, соответствующий ожидаемому поведению системы.

Описанный аппарат удобен для анализа существующих последовательностей событий в ВС (например, для анализа трасс), но не удобен для моделирования функционирования ВС, так как для некоторой локации и идущего из нее перехода нельзя указать, когда данный переход *должен* быть совершен (переход размечен лишь условием, при выполнении которого переход *может* быть совершен). Для преодоления данного недостатка аппарат временных автоматов был модифицирован в работе [85]. Было предложено каждой локации автомата поставить в соответствие инвариант — ограничение на значения таймеров. Некоторая локация может быть текущей только если соответствующий инвариант выполняется. Значения всех таймеров могут быть увеличены на некоторое положительное вещественное значение d, если для любого $d' \leqslant d$ текущие значения таймеров, увеличенные на d', удовлетворяют инварианту текущей локации. Такое увеличение значений таймеров называется продвижением времени. Если некоторая локация является текущей для автомата, и значения таймеров не могут быть увеличены ни на какое положительное значение, то либо должен быть совершен переход из данной локации, либо, если ни один переход не возможен, фиксируется состояние тупика.

В работе [86] описано дальнейшее расширение этого формализма. С помощью оператора параллельной композиции (|), введенного в исчислении взаимодействующих систем (Calculus of Communicating Systems) Р. Милнера [87], временные автоматы было предложено объединять в сети. Взаимодействие автоматов осуществляется посредством разделяемых целочисленных переменных, а также посредством синхронизаций по каналам. Состояние сети временных автоматов определяется совокупностью текущих локаций автоматов, значений переменных и таймеров.

Каждая переменная имеет конечное число возможных значений. Значения переменных могут использоваться в условиях переходов, а также изменяться при выполнении переходов.

Каждый переход автомата размечен символом алфавита, соответствующего всем возможным вариантам синхронизаций по каналам. Каждому каналу соответствуют два символа алфавита, обозначающих соответственно отправку и прием сигнала по данному каналу. Также в алфавите

присутствует специальный символ, обозначающий отсутствие синхронизации. Если в некотором состоянии сети автоматов в одном из автоматов сети может быть выполнен переход, помеченный отправкой сигнала по каналу, а в другом автомате сети — приемом сигнала по тому же каналу, то оба перехода выполняются как единое действие. Если в некотором состоянии сети автоматов в нескольких автоматах сети могут быть выполнены переходы, помеченные отправкой сигнала по каналу, а в нескольких автоматах — приемом сигнала по тому же каналу, то пара автоматов для синхронизации выбирается недетерминированным образом, так чтобы в одном из выбранных автоматов был возможен переход, помеченный отправкой сигнала по каналу, а в другом — приемом сигнала по тому же каналу. Кроме того, существуют широковещательные каналы. Если в некотором состоянии сети автоматов в одном из автоматов сети может быть выполнен переход, помеченный отправкой сигнала по широковещательному каналу, а в нескольких других автоматах — приемом сигнала по тому же каналу, все такие переходы выполняются как единое действие. Если в некотором состоянии сети автоматов в нескольких автоматах сети могут быть выполнены переходы, помеченные отправкой сигнала по широковещательному каналу, то автомат, в котором осуществляется отправка сигнала по широковещательному каналу, выбирается недетерминированным образом.

Любая сеть временных автоматов с помощью известного алгоритма [86] может быть преобразована в один временной автомат.

Каждая возможная последовательность состояний сети называется ее *вычислением.* Если модель МВС построена в виде сети временных автоматов, то по вычислению этой сети автоматов может быть построена ВД модели, то есть данный математический аппарат удовлетворяет требованию 1 раздела 2.1.1.

Данный математический аппарат позволяет проверять выполнимость формул в логике TCTL (расширение темпоральной логики CTL [88] для работы с временем как количественной величиной, описанное в [86]). Пусть φ — некоторое логическое выражение, значение которого для любого состояния сети автоматов определяется однозначно и не зависит от других состояний. В формулу φ могут входить в том числе и ограничения на значения таймеров. Логика TCTL позволяет формулировать свойства следующего вида:

- $A[]\varphi$ — для всех вычислений верно, что во всех состояниях вычисления выполняется φ;
- $E <> \varphi$ — существует вычисление, в некотором состоянии которого выполняется φ;
- $A <> \varphi$ — для всех вычислений верно, что существует состояние вычисления, в котором выполняется φ;
- $E[]\varphi$ — существует вычисление, во всех состояниях которого выполняется φ;
- $\varphi \rightarrow \psi$ — для всех вычислений верно, что если в некотором состоянии вычисления выполнено φ, то в этом вычислении существует последующее состояние, в котором будет выполнено ψ (при этом между состоянием, в котором выполнено φ, и состоянием, в котором выполнено ψ, может быть конечное число других состояний).

Существуют алгоритмы [86] проверки выполнения свойств перечисленного вида для заданной сети автоматов.

Математический аппарат сетей временных автоматов с темпоральной логикой TCTL использовался в ряде проектов, посвященных моделированию и верификации промышленных

вычислительных систем. Так в [21] с помощью этого математического аппарата была описана модель поведения бортового компьютера автомобиля и модель многопроцессорной бортовой вычислительной системы. В [89] предложена модель, описывающая функционирование планировщика одной из операционных систем реального времени. Требования к моделям в этих работах были сформулированы как TCTL-формулы.

Таким образом, данный математический аппарат удовлетворяет требованию 3 раздела 2.1.1.

Примерами инструментальных средств для построения и верификации сетей временных автоматов являются UPPAAL [90] и IMITATOR [91]. При этом IMITATOR позволяет проверять выполнимость TCTL-формул лишь двух из пяти приведенных выше видов: $A[]\varphi$ и $A <> \varphi$ [92]. UPPAAL позволяет проверять выполнимость TCTL-формул всех указанных видов. Следовательно, 1 выполняется требование 4 раздела 2.1.1.

Недостатком формализма сетей временных автоматов является невозможность моделирования с его помощью вытеснения работ с ядра процессора [63, 93]. Для моделирования вытеснения работ математический аппарат должен позволять описывать следующее поведение работы: работа может находиться в нескольких состояниях, в том числе в состоянии E (соответствует выполнению работы на ядре). Работа может покидать состояние E и возвращаться в него, при этом в состоянии E работа должна находиться суммарно n единиц времени (максимальная длительность выполнения работы на ядре процессора). Предположим, что поведение работы моделируется с помощью временного автомата, имеющего локацию E, соответствующую состоянию работы E. Математический аппарат временных автоматов не позволяет присваивать переменным и таймерам значения других таймеров, и значения всех таймеров увеличиваются синхронно. Поэтому невозможно сохранить значение некоторого таймера (равное, например $n_1 < n$) в момент перехода из локации E с тем, чтобы при возвращении в эту локацию таймер продолжил увеличение, начиная со значения n_1. В работе [17] с использованием аппарата сетей временных автоматов построена модель планировщика, поддерживающего вытеснение. Однако для моделирования времени в этой модели используются обычные целочисленные переменные, что приводит к недостаткам, описанным выше на странице 27.

Таким образом, требование 2 раздела 2.1.1 для сетей временных автоматов выполняется лишь частично. Чтобы преодолеть описанную проблему, в работе [94] были предложены расширения математического аппарата временных автоматов, описанные далее.

2.1.3. Сети временных автоматов с остановкой таймеров

В работе [94] было предложено расширение аппарата сетей временных автоматов, заключающееся в использовании при моделировании таймеров с возможностью остановки (stopwatches) вместо обычных таймеров (timers). Для каждой пары ⟨локация, таймер⟩ задается условие активности таймера в этой локации — функция над множеством переменных, возвращающая логическое значение. Если для текущей локации условие активности истинно, то соответствующий таймер

считается активным, иначе — остановленным. Продвижение времени в локации выполняется только для активных в ней таймеров.

В [94] показано, что в общем случае математический аппарат сетей временных автоматов с остановкой таймеров эквивалентен по выразительности аппарату линейных гибридных автоматов [79], что означает алгоритмическую неразрешимость большинства задач верификации, в том числе задачи проверки достижимости заданного состояния. Поэтому на практике используются отдельные классы сетей временных автоматов с остановкой таймеров, например с ограниченным (не более двух в общем случае, не более семи с дополнительными ограничениями на вид автоматов) числом таймеров [95], с ограничениями на значения времени [96].

В ряде работ [97—99] показано, что математического аппарата сетей временных автоматов с остановкой таймеров достаточно для моделирования планировщиков, допускающих вытеснение. При этом доказано, что для построенных авторами моделей задача достижимости некоторого состояния алгоритмически разрешима.

Средства UPPAAL [90] и IMITATOR [91] предоставляют возможность моделирования и верификации сетей автоматов с остановкой таймеров.

Таким образом, данный математический аппарат удовлетворяет всем требованиям раздела 2.1.1.

2.1.4. Временные сети Петри

Данный математический аппарат представляет собой расширение аппарата сетей Петри [100]. Графически сеть Петри может быть представлена в виде двудольного ориентированного мультиграфа с двумя типами вершин: позиции и переходы. Дуга графа может вести только из позиции в переход, либо из перехода в позицию. Для сети Петри задается маркировка: каждой позиции ставится в соответствие неотрицательное число фишек. Некоторый переход может быть выполнен, если каждая из его входных позиций содержит по крайней мере столько фишек, сколько имеется дуг, ведущих из данной позиции в данный переход. При выполнении переходов фишки во входных позициях, соответствующие ведущим в переход дугам, удаляются. В выходных позициях создаются новые фишки, по одной на каждую дугу, ведущую из выполняемого перехода. Если текущая маркировка сети Петри такова, что возможно выполнение нескольких переходов, то выбор перехода для выполнения осуществляется недетерминированно. Состояние сети Петри определяется ее маркировкой. При моделировании некоторых систем (в том числе вычислительных, например в [101]) сетями Петри переходам, как правило, соответствуют события системы, позициям — условия возникновения таких событий.

Каждая возможная последовательность состояний сети Петри называется ее вычислением.

Как правило, при верификации сетей Петри решаются следующие задачи:

– проверка достижимости позиции (существует ли такое вычисление, на котором в позиции появляется хотя бы одна фишка);

– проверка ограниченности (для всех ли вычислений количество фишек в позициях ограничено);
– проверка достижимости определенной маркировки.

Первое временное расширение описанного формализма было предложено в работе [102]. Во временных сетях Петри каждый переход помечен двумя неотрицательными вещественными числами a и b, $a \leqslant b$, задающими минимальную и максимальную задержку выполнения перехода. Переход должен быть выполнен через t единиц времени после того, как во всех входных позициях появится необходимое количество фишек, где $a \leqslant t \leqslant b$. То есть с каждым возможным при данной маркировке переходом связан один вещественный таймер. Как только во всех входных позициях перехода появится необходимое количество фишек соответствующий таймер активируется, его значение при этом равно 0. Состояние временной сети Петри определяется ее маркировкой, а также значениями таймеров, связанных с возможными для данной маркировки переходами. Значения таймеров, связанных с невозможными при данной маркировке переходами, не определены. Значения всех таймеров, соответствующих возможным переходам, могут быть согласованно увеличены на некоторое положительное вещественное значение d, если для любого $d' \leqslant d$ текущие значения всех таймеров, увеличенные на d', не превосходят правых границ соответствующих интервалов.

Другим известным временным расширением аппарата сетей Петри являются сети Петри с временными дугами [103]. Каждой фишке сопоставлен вещественнозначный таймер, который запускается в момент появления фишки в позиции. Каждая дуга, ведущая в переход, помечена интервалом. Переход может быть выполнен, если для каждой его входной дуги в соответствующей входной позиции имеется фишка, значение таймера которой принадлежит интервалу, указанному для дуги. Для сети Петри с временными дугами состояние определяется ее маркировкой, а также значениями таймеров, связанными с фишками данной маркировки. Дальнейшее расширение данного формализма [104] позволяет задать для перехода максимальное значение для таймеров, соответствующих фишкам входных позиций: если маркировка и ограничения на дугах допускают переход и хотя бы одно из значений таймеров достигло максимального значения, то переход *обязан* быть выполнен. Значения всех таймеров могут быть согласованно увеличены на некоторое положительное вещественное значение d, если для любого $d' \leqslant d$ текущие значения таймеров, увеличенные на d', не превосходят соответствующих максимальных значений.

По вычислению временной сети Петри или сети Петри с временными дугами можно построить ВД моделируемого объекта (требование 1 раздела 2.1.1).

В общем случае для временных сетей Петри перечисленные выше задачи верификации алгоритмически неразрешимы [104], поэтому на практике используют ограниченные временные сети Петри, количество фишек в позициях которых ограничено. Для верификации таких сетей Петри существует ряд инструментальных средств (например, [105, 106]), однако указанные средства не позволяют проверять свойства, время в которых используется как количественная величина. Примером такого свойства может быть следующее: «Существует такое вычисление сети Петри, что переход T выполняется более, чем за 5 единиц времени». Для проверки подобных свойств выполняется преобразование временной сети Петри во временной автомат, и необходимое свойство

проверяется для него посредством верификатора UPPAAL. В работе [107] приведен алгоритм такого преобразования, а также доказано, что ограниченные временные сети Петри и временные автоматы имеют одинаковую выразительность.

Для сетей Петри с временными дугами задачи проверки достижимости позиции и ограниченности алгоритмически разрешимы, в отличие от задачи проверки достижимости маркировки [104]. В работе [108] аппарат сетей Петри с временными дугами был доработан для возможности описания достаточных условий выполнения переходов. Для верификации таких сетей Петри необходимо их преобразование во временные автоматы, которые затем анализируются в средстве UPPAAL [90].

Таким образом, требования 3 раздела 2.1.1 выполняется для временных сетей Петри с дополнительными ограничениями. Требование 4 выполняется, но для решения некоторых задач моделирования и верификации используется преобразование временных сетей Петри во временные автоматы.

Временные сети Петри и сети Петри с временными дугами используются для моделирования функционирования вычислительных систем (например, в работах [108, 109]). Однако, аналогично временным автоматам, этот математический аппарат не позволяет моделировать вытеснение работ. То есть требование 2 раздела 2.1.1 также выполняется лишь частично.

2.1.5. Временные сети Петри с остановкой таймеров

Для моделирования планировщиков, допускающих вытеснение, с помощью временных сетей Петри необходим механизм остановки таймеров (аналогично ситуации с временными автоматами). Такое расширение описано в работе [110].

Авторами этой работы было предложено выделить два типа переходов: прерываемые и непрерываемые. Каждому переходу сопоставлен таймер. При выполнении перехода значение соответствующего таймера обнуляется. Если выполнение перехода t_1 приводит к тому, что новая маркировка не допускает другой переход t_2, то имеют место три варианта:

1. если и старая маркировка не допускала t_2, то значение таймера, сопоставленного t_2, не изменяется;
2. если старая маркировка допускала t_2 и переход t_2 прерываемый, то таймер, сопоставленный t_2, помечается как остановленный, а его значение сохраняется; как только маркировка сети станет вновь допускать t_2, соответствующий таймер будет помечен как активный, а его значение начнет увеличиваться, начиная с последнего сохраненного значения;
3. если старая маркировка допускала t_2 и переход t_2 непрерываемый, то как только маркировка сети станет вновь допускать t_2, значение соответствующего таймера начнет увеличиваться, начиная с нулевого значения.

Правила увеличения значений таймеров совпадают с правилами для обычных временных сетей Петри.

Верификация описанных в [110] сетей Петри осуществляется посредством их преобразования во временные автоматы с остановкой таймеров и использования методов и средств верификации таких автоматов.

Таким образом, временные сети Петри с остановкой таймеров удовлетворяют всем требованиям раздела 2.1.1.

2.1.6. Алгебра процессов с временем

Еще одним способом описания распределенных ВС является алгебра взаимодействующих процессов [87, 111, 112]. Этот подход предполагает представление процессов, имеющих место в ВС, в виде формул, составленных по определенным правилам. Вычисление системы взаимодействующих процессов выражается в выполнении действий. Действия могут быть внутренними и внешними (по отношению к некоторому процессу). Внутренние действия процесса затрагивают только этот процесс, внешние действия выполняются только согласованно с внешними действиями других процессов.

Для процессов определен ряд операций. Состав этих операций зависит от конкретной алгебры процессов, но, как правило, во всех алгебрах процессов определены следующие операции [112]:

- **префиксное действие «.»**. Если P_1, P_2 — некоторые процессы, то $P_1.P_2$ — процесс, всякое вычисление которого начинается вычислением процесса P_1 и по завершении его продолжается вычислением процесса P_2;
- **альтернативная композиция «+»**. Если P_1, P_2 — некоторые процессы, то $P_1 + P_2$ — процесс, всякое вычисление которого является либо вычислением процесса P_1, либо вычислением процесса P_2;
- **параллельная композиция «|»**. Если P_1, P_2 — некоторые процессы, то $P_1|P_2$ — процесс, всякое вычисление которого представляет собой чередование вычислений процессов P_1 и P_2, сопровождаемое совместным выполнением внешних действий.

Понятие процесса вводится рекурсивным образом [112]. Процесс — это всякое выражение, которое может быть построено по следующим правилам:

- 0 — нулевой процесс;
- если P — процесс, а n — действие, то $n.P$ — процесс; в частности, $n.0$ — процесс из одного действия;
- если P_1, P_2 — процессы, то $P_1 + P_2$, $P_1|P_2$ — процессы;
- других процессов нет.

Для введенных операций определен ряд свойств. Набор этих свойств, как и набор операций, специфичен для конкретной алгебры процессов. Приведем примеры свойств, как правило, определенных в алгебрах процессов:

- операция «+» коммутативна, т.е. $\forall P_1, P_2 :\ P_1 + P_2 = P_2 + P_1$;
- 0 является нейтральным элементом относительно операции «|», т.е. $\forall P :\ P|0 = P$;

– операция «|» ассоциативна, т.е. $\forall P_1, P_2, P_3 : (P1|P2)|P3 = P1|(P2|P3)$.

Также в алгебре процессов вводится понятие эквивалентности процессов (обозначается символом «$\backsim$») и доказывается ряд его свойств, например:

– если $P_1 \backsim P_2$, то $\forall P : P_1 + P \backsim P_2 + P$;

– $\forall P : P + P \backsim P$;

– если $P_1 \backsim P_2$, то $\forall P : P_1|P \backsim P_2|P$;

Посредством применения свойств операций и свойств эквивалентности для процессов выполняются эквивалентные преобразования, позволяющие упростить процесс, либо привести его к некоторому эталонному виду.

Согласно [111], процесс может быть представлен как размеченный направленный граф, вершины которого соответствуют состояниям моделируемой системы, а ребра — переходам между этими состояниями с выполнением действий. Одно из состояний процесса помечено как начальное. Нулевым процессом (0) считается процесс, состоящий из одного состояния и не имеющий переходов.

Существуют следующие подходы к верификации процессов:

– доказательство эквивалентности заданного процесса, либо процесса, полученного из данного посредством применения операции ограничения (ограничение процесса P по множеству L — процесс, полученный из P посредством удаления тех переходов, которые размечены действиями из L) некоторому эталонному процессу путем эквивалентных преобразований [111];

– формализация требований к системе в виде формул темпоральных логик (LTL, CTL и т.д.) и проверка их выполнения с помощью верификаторов, работающих с размеченными системами переходов [112].

Существует ряд алгебр процессов, позволяющих работать с временем как с количественной величиной: временные расширения алгебр ACP [113], CSS [114], CSP [115] и др. Однако, эти математические аппараты изначально не предназначены для моделирования вычислительных систем, в которых используется совместно несколько различных алгоритмов планирования (в т.ч. планирование согласно статическому расписанию и динамическое планирование с вытеснением).

Рассмотрим в качестве примера алгебру процессов PACOR (Process Algebra of Communicating Resources) [116]. Класс вычислительных систем, для моделирования которых предназначена эта алгебра процессов, наиболее близок (по сравнению с другими алгебрами процессов) к МВС, рассматриваемым в данной работе. В PACOR для процесса задается набор ресурсов. Действия данной алгебры делятся на два вида:

– *Требующие ресурсов*. Такие действия описываются с помощью конструкций вида: $\{(res, prio)\}^{[min,max]}$, либо $\langle(res, prio)\rangle^{[min,max]}$, где res — идентификатор ресурса, $prio$ — приоритет процесса, min,max — минимальная и максимальная длительность использования ресурса. Скобки $\{\}$ обозначают, что данное действие может быть прервано в связи с необходимостью доступа к ресурсу другого действия, а после освобождения ресурса — возобновлено. Скобки $\langle\rangle$ означают, что прерывание действия запрещено.

– *Не требующие ресурсов*.

Также для действия можно указать директивный срок выполнения и процесс, на который должно переключиться выполнение в случае нарушения этого директивного срока.

Отметим, что этот формализм хорошо подходит для моделирования функционирования динамических планировщиков с фиксированными приоритетами. Однако описание других алгоритмов планирования может потребовать доработки формализма и привести к чрезмерному усложнению моделей. Так, алгебра процессов PACOR не предполагает динамического изменения приоритетов процессов, которое имеет место, например, при планировании вычислений по стратегии EDF: на выполнение ставится та работа из множества готовых работ, правая граница директивного интервала которой в данный момент является наименьшей (то есть приоритет работы зависит от близости правой границы ее директивного интервала к текущему моменту времени).

Таким образом, требование 2 раздела 2.1.1 для PACOR выполняется частично. Согласно [111], в алгебрах процессов существует понятие трассы, близкое к понятию вычисления сети временных автоматов. По такой трассе можно построить ВД. Следовательно, требование 1 выполняется. Выше указаны основные подходы к верификации процессов (требование 3). Для анализа моделей в PACOR используется их преобразование в сети временных автоматов с остановкой таймеров и дальнейшая верификация с использование верификатора UPPAAL. В работе [116] приведен алгоритм такого преобразования, однако в открытых источниках не найдено программное средство, реализующее этот алгоритм. То есть требование 4 выполняется частично.

2.1.7. Модель с сообщениями

В работах [73, 117] предложена модель функционирования распределенной вычислительной системы с временем, носящая название модели с сообщениями (МС-модели). В отличие от описанных ранее математических аппаратов, позволяющих моделировать объекты различной природы, МС-модель ориентирована на моделирование распределенных вычислительных систем реального времени (РВС РВ).

Основу МС-модели составляют следующие понятия: распределенный исполнитель — модель аппаратных средств, поведение — модель функционирования программного обеспечения, и наблюдатель.

Распределенный исполнитель представляет собой совокупность последовательных исполнителей, соединенных каналами связи. Последовательный исполнитель может соответствовать процессору либо вычислительному ядру моделируемой системы. Для последовательного исполнителя задана длительность его такта (соответствует астрономическому времени такта моделируемого процессора). Также для последовательного исполнителя определен набор атомарных процессов, которые он может выполнять. Для каждого атомарного процесса задан интервал, которому принадлежит длительность его выполнения в тактах. В каждый момент времени последовательный исполнитель может выполнять не более одного атомарного процесса. Атомарные процессы не прерываемы. Кроме того, для последовательного исполнителя задан набор входов и выходов. Каждому входу и выходу однозначно соответствует один из процессов (понятие

процесса будет введено далее), причем выходу соответствует атомарный процесс последовательного исполнителя. Каждый выход последовательного исполнителя связан каналом связи с входом другого или того же самого последовательного исполнителя. Для каждого такта времени работы последовательного исполнителя определена функция возбуждения входов, ставящая в соответствие каждому входу 0 или 1. Если для некоторого входа эта функция равна 1, такой вход называется возбужденным. Аналогичным образом определена функция возбуждения выходов. Если для заданного такта времени некоторый выход последовательного исполнителя возбужден, то в течение этого такта на этом исполнителе выполняется атомарный процесс, соответствующий возбужденному выходу. Логика возбуждения входов и выходов реализована в последовательном наблюдателе, описанном ниже. Передача возбуждения с входов на выходы реализована в наблюдателе за дугой, также описанном ниже. Для последовательного исполнителя задана функция-арбитр, осуществляющая выбор некоторого входа из множества возбужденных входов. Автором работы [117] была разработана алгебра распределенных исполнителей, позволяющая моделировать РВС РВ произвольной структуры.

Для всей модели задано множество сообщений, передаваемых по каналам связи, то есть по связям между выходами и входами последовательных исполнителей. Для каналов известна пропускная способность, а для каждого сообщения — размер.

Ключевым понятием модели функционирования программного обеспечения в рамках МС-модели является процесс. Каждый процесс характеризуется ациклическим (возможно, бесконечным) ориентированным графом своего поведения. Каждая вершина графа соответствует состоянию процесса. Одна из вершин, в которую не входит ни одной дуги, является корневой вершиной. Каждой вершине графа поведения процесса, кроме корневой вершины, соответствует шаг процесса. Шаг процесса характеризуется кортежем из четырех элементов: сообщение-воздействие, последовательность внутренних действий, сообщение-реакция и процесс, которому данный шаг передает управление и сообщение-реакцию. Каждому внутреннему действию сопоставлена последовательность атомарных процессов того последовательного исполнителя, к которому привязан процесс. Произвольный (возможно, бесконечный) путь в графе поведения называется историей процесса. Граф поведения всей системы представляет собой совокупность графов поведения всех процессов системы.

Функционирование системы заключается в формировании историй всех ее процессов. В начальный момент времени управление передано некоторому фиксированному (определенному при построении модели) процессу. После выполнения шага процесса управление передается либо процессу, указанному в описании этого шага, либо (если текущий процесс атомарный) процессу, вызвавшему текущий процесс. Формирование истории процессов выполняет распределенный наблюдатель, представляющий собой совокупность последовательных наблюдателей (соответствующих последовательным исполнителям) и наблюдателей за дугами.

Последовательный наблюдатель — это недетерминированный стековый автомат с несколькими входами и выходами, входы и выходы которого взаимно однозначно соответствуют входам и выходам соответствующего последовательного исполнителя.

Входные сигналы (т.е. входной алфавит) последовательного наблюдателя принадлежат множеству пар вида (сообщение, состояние процесса), объединенному со специальным сигналом «*».

Подача элемента (m, v) на вход i последовательного наблюдателя означает, что i-й вход последовательного исполнителя возбужден и на него поступило сообщение m, инициатором которого был шаг, соответствующий состоянию v некоторого процесса. Подача сигнала «*» на некоторый вход последовательного наблюдателя обозначает, что соответствующий вход последовательного исполнителя не возбужден. Аналогичным образом определяются выходные сигналы (выходной алфавит) последовательного наблюдателя.

Состояние последовательного наблюдателя определяется номером возбужденного входа последовательного исполнителя, текущим процессом на этом исполнителе и состоянием этого процесса. При выполнении перехода между состояниями в последовательном наблюдателе текущим входным сигналам, текущему состоянию и текущей вершине стека ставятся в соответствие выходные сигналы, новое состояние и новая вершина стека. Такой переход соответствует смене текущего состояния процесса (и, возможно, самого текущего процесса) в последовательном исполнителе и возбуждению его выходов с поступлением на них сообщений. При этом смена текущего состояния процесса (и, возможно, самого текущего процесса) может происходить как в результате внешних воздействий (т.е. возбуждения входов и появления на них сообщений), так и в результате выполнения шагов процессов, и сопровождающуюся возбуждением выходов и формированием на них сообщений. Если у последовательного исполнителя возбуждены несколько входов, то при переходе в последовательном наблюдателе выбирается процесс, соответствующий тому входу, который выберет функция-арбитр исполнителя. Если возбужденных входов нет, то выбор следующего шага осуществляется исходя из текущего состояния текущего процесса.

Наблюдатель за дугой — это недетерминированный автомат, аналогичный автомату-последовательному наблюдателю, но не имеющий памяти (стека). Этот автомат моделирует передачу сообщений по каналам связи, то есть передачу возбуждения с выхода последовательного исполнителя на соединенный с этим выходом вход последовательного исполнителя (другого или того же самого).

В формализме МС-моделей имеется понятие временной диаграммы, отличное от понятия временной диаграммы, используемого в данной диссертационной работе. Под временной диаграммой понимается набор функций (по одной функции на каждый последовательной исполнитель), каждая из которых ставит в соответствие заданному значению астрономического времени четверку (имя текущего процесса, имя текущего шага процесса, имя текущего внутреннего действия в этом шаге, имя текущего атомарного процесса). Имея такую временную диаграмму, можно построить временную диаграмму, соответствующую определению, введенному в разделе 1.4.2, необходимую для проверки выполнения ограничений реального времени. Таким образом, требование 1 раздела 2.1.1 выполняется.

В работе [117] доказано, что МС-модель превосходит по выразительности конечные автоматы, но имеет меньшую выразительность, чем сети Петри.

Также в [117] введено понятие источников сообщений с временем. Источник задает конечную или бесконечную последовательность сообщений, каждому из которых приписана (детерминированно или случайно) величина задержки в виде количества тактов исполнителя. Процессы могут обращаться к источникам с запросами (такой запрос — одно из возможных внутренних действий процесса). Длительность выполнения запроса равна величине задержки, приписанной

сообщению. Источники сообщений с временем позволяют работать с временем как с количественной величиной. В работе [118] приведено описание основных компонентов МС-моделей (с некоторыми ограничениями) в математическом аппарате временных автоматов, а также подходов к их верификации (требование 3 выполняется).

Наличие в МС-модели атомарных (непрерываемых) процессов затрудняет моделирование планировщиков с вытеснением. При проектировании МВС с такими планировщиками считается, что работа (процесс) может быть вытеснена (прервана) в любой момент времени, если того потребует выбранный алгоритм планирования. Поэтому при использовании формализма МС-моделей для моделирования таких МВС потребуется задавать атомарные процессы с очень малыми длительностями выполнения (например, равными наибольшему общему делителю тактов исполнителя), что приведет к чрезмерному усложнению моделей. Наличие единственной функции-арбитра для последовательного исполнителя затрудняет моделирование использования различных планировщиков в разных разделах, привязанных к одному ядру. Таким образом, требование 2 для МС-моделей выполняется частично.

На основе формализма МС-моделей было создано несколько программных средств для моделирования и верификации РВС РВ, однако в настоящее время большая их часть не поддерживается (требование 4 выполняется частично).

2.1.8. Выводы

В таблице 2.1 приведены результаты выполненного обзора математических аппаратов для моделирования функционирования МВС. В таблице используются следующие сокращения: TA (Timed Automata) — временные автоматы, SA (Stopwatch Automata) — временные автоматы с остановкой таймеров, TPN (Timed Petri Nets) — временные сети Петри, SPN (Stopwatch Petri Nets) — сети Петри с остановкой таймеров, PA (Process Algebra) — алгебры процессов, МС — МС-модель.

Таблица 2.1 — Соответствие математических аппаратов требованиям раздела 2.1.1

№	Описание требования	Математические аппараты					
		TA	SA	TPN	SPN	PA	MC
1	Возможность получения ВД, соответствующей модели и содержащей события с их временными метками	+	+	+	+	+	+
2	Возможность моделирования всех существенных аспектов функционирования МВС	±	+	±	+	±	±
3	Возможность проверки корректности моделей математическими методами	+	+	+	+	+	+
4	Наличие инструментального ПО для разработки, прогона и верификации моделей	+	+	+	+	±	±

Обзор показал, что всем предъявленным в разделе 2.1.1 требованиям соответствуют временные автоматы (и их сети) с остановкой таймеров и сети Петри с остановкой таймеров. Как было отмечено в разделе 2.1.5, для верификации моделей, описанных в виде сетей Петри с остановкой таймеров, используется преобразование этих моделей во временные автоматы с остановкой таймеров и последующая верификация таких автоматов. Для того, чтобы избежать подобных дополнительных преобразований, в качестве математического аппарата для моделирования функционирования МВС автором были выбраны сети временных автоматов с остановкой таймеров.

2.2. Формальное определение сетей временных автоматов с остановкой таймеров

Кратко сети временных автоматов с остановкой таймеров были описаны в разделе 2.1.3. В настоящем разделе приведено строгое определение данного математического аппарата, основанное на [86, 90, 94]. В указанных источниках некоторые понятия введены неформально, поэтому соответствующие им строгие определения были введены автором.

Обозначим символом V конечный упорядоченный набор целочисленных переменных, $\overline{v} = (v_1, ..., v_{|V|})$ — вектор значений этих переменных. Каждая переменная может принимать конечное число значений. Для целочисленных переменных определены стандартные арифметические операции над целыми числами (сложение, вычитание, умножение, деление нацело, взятие остатка от деления нацело), а также операции сравнения ($=, \neq, <, >, \leqslant, \geqslant$) и присваивания. В перечисленных операциях, наряду с переменными, могут использоваться целочисленные константы (кроме левой части операции присваивания).

Таймером назовем переменную, принимающую неотрицательные вещественные значения, для которой определены операция сравнения с целочисленными выражениями и операция обнуления. Элементарное сравнение таймеров — предикат вида $x\ op\ n$ или $(x - y)\ op\ n$, где x,y — таймеры, n — целочисленное выражение, op — одна из операций сравнения. В данной работе для упрощения верификации выражения вида $(x - y)\ op\ n$ использоваться не будут. Обозначим символом C конечный упорядоченный набор таймеров, $\overline{c} = (c_1, ..., c_{|C|})$ — вектор значений этих таймеров.

Предусловием назовем логическое выражение вида B_1, B_2, $(B_1) \wedge (B_2)$, или $(B_1) \vee (B_2)$, где B_1 — произвольное логическое выражение над сравнениями выражений, включающих целочисленные переменные из V и константы, и не содержащих операций присваивания, B_2 — элементарное сравнение таймеров, либо конъюнкция их элементарных сравнений. $B(C,V)$ — множество всевозможных предусловий над таймерами и переменными.

Под конечным временным автоматом с остановкой таймеров будем понимать кортеж $A = \langle L, l_0, C, V, \overline{v_0}, AA, AS, E, I, P, U', U'' \rangle$, задающий направленный размеченный граф (возможно, с петлями и кратными дугами), где

– L — множество локаций автомата, являющееся множеством вершин графа.

- l^0 — начальная локация, $l^0 \in L$.
- C — конечный упорядоченный набор таймеров с нулевыми начальными значениями.
- V — конечный упорядоченный набор переменных.
- $\overline{v_0}$ — начальные значения всех переменных.
- AA — множество действий по обнулению таймеров вида $c_i := 0$ и по изменению значений переменных вида $v_i := f_i(\overline{v})$, где $f_i(\overline{v})$ — целочисленное выражение, содержащее переменные и константы; в AA также входит элемент ε, обозначающий отсутствие действий; $F(AA)$ — множество всех упорядоченных наборов из элементов AA, один такой набор может содержать несколько экземпляров действия из AA (например, одна и та же переменная может быть увеличена на 1 несколько раз).
- AS — множество действий по синхронизации автоматов посредством каналов. Семантика действий синхронизации и условия возможности их выполнения будут введены далее при формализации поведения сети автоматов. Специальное действие $\tau \in AS$ обозначает отсутствие действия синхронизации.
- E — множество переходов, то есть размеченных дуг, ведущих из одной локации в другую, $E \subseteq L \times B(C,V) \times AS \times F(AA) \times L$. Пусть $e =< l, b, as, aa, l' >, e \in E$ — дуга из локации l в локацию l' (далее будет обозначаться как $l \xmapsto{b,as,aa} l'$). Наличие данной дуги означает, что если предусловие b истинно при текущих значениях таймеров и переменных и возможно выполнение действия синхронизации $as \in AS$, то возможен переход из локации l в локацию l'. При этом выполняются действия $aa \in F(AA)$ (действия выполняются в последовательности, определяемой aa; с точки зрения внешнего наблюдателя последовательность действий неделима).
- $I : L \to B(C,V)$ — назначение инвариантов. Инвариант $I(l)$, приписанный локации l — необходимое условие, которое должно быть соблюдено для того, чтобы автомат мог находиться в локации l.
- $P : (L \times C) \to B(\varnothing, V)$ — назначение условий активности таймеров. Если для некоторого таймера, текущей локации автомата и текущих значений переменных условие активности выполнено, то значение этого таймера может быть увеличено. В противном случае увеличение значения этого таймера невозможно. Все правила увеличения значений таймеров (то есть продвижения времени) будут описаны далее.
- $U' \subseteq L$ — множество срочных локаций, то есть локаций, в которых запрещено увеличение значения какого-либо таймера.
- $U'' \subseteq U'$ — множество приоритетных локаций. Эти локации являются срочными и для них верно, что если в одном из автоматов сети автоматов текущая локация является приоритетной, то локация-источник следующего перехода в сети автоматов должна быть срочной. Согласно выбранному подходу к верификации автоматов-моделей компонентов МВС (см. раздел 3.2.1), в данной работе *запрещено* использовать приоритетные локации в качестве *начальных* локаций в автоматах-моделях компонентов МВС.

Далее под понятием «автомат» понимается временной автомат с остановкой таймеров, а под понятием «сеть автоматов» — сеть временных автоматов с остановкой таймеров.

Согласно [86], записью $(\overline{c},\overline{v}) \in I(l)$, где l — некоторая локация автомата, обозначается, что для значений $\overline{c}$ всех таймеров и значений $\overline{v}$ всех переменных инвариант $I(l)$ выполняется. Аналогично, $(\overline{c},\overline{v}) \in b$ означает, что для $(\overline{c},\overline{v})$ выполняется предусловие b; $\overline{v} \in P(l,x)$ — для значений переменных $\overline{v}$ выполняется условие активности таймера x в локации l.

Состоянием автомата называется тройка $(l,\overline{c},\overline{v})$, задающая локацию автомата, значения его таймеров и переменных. $S = L \times \mathbb{R}_{\geqslant 0}^{|C|} \times \mathbb{Z}^{|V|}$ — множество состояний автомата.

Поведение автомата описывается системой переходов $(S, s_0, \mapsto)$, где $s_0 = (l_0, \overline{0}, \overline{v_0})$ — начальное состояние, $\mapsto \subseteq ((S \times \mathbb{R}_{\geqslant 0} \times S) \cup (S \times (AS \times F(AA)) \times S))$ — отношение перехода. Переходы могут быть дискретными (то есть принадлежать множеству $(S \times (AS \times F(AA)) \times S)$) и непрерывными (то есть принадлежать множеству $S \times \mathbb{R}_{\geqslant 0} \times S$). Опишем, в каких случаях определены переходы каждого типа для заданного автомата:

1. определен дискретный переход $(l,\overline{c},\overline{v}) \overset{as,aa}{\longmapsto} (l',\overline{c'},\overline{v'})$, если существует дуга $(l \overset{b,as,aa}{\longmapsto} l') \in E : (\overline{c},\overline{v}) \in b, (\overline{c'},\overline{v'}) \in I(l')$, а $(\overline{c'},\overline{v'})$ получаются из $(\overline{c},\overline{v})$ посредством выполнения действий aa. То есть происходит смена локации автомата с обнулением некоторых таймеров и изменением переменных за счет выполнения действий из aa, при которой новые значения таймеров и переменных должны удовлетворять инварианту нового состояния.
2. для некоторого $d \in \mathbb{R}_{>0}$ определен непрерывный переход
   $(l,\overline{c},\overline{v}) \overset{\tau,\varepsilon}{\longmapsto} (l, inc(\overline{c},d,l,\overline{v}),\overline{v})$, если выполняется каждое из следующих условий:
   а) $l \notin U'$;
   б) $\forall d' \in \mathbb{R}_{\geqslant 0} : d' \leqslant d$ справедливо соотношение $(inc(\overline{c},d',l,\overline{v}),\overline{v}) \in I(l)$

   То есть локация автомата остается прежней, и происходит увеличение значений таймеров, при котором инвариант локации должен оставаться истинным.
   Функция $inc : \mathbb{R}_{\geqslant 0}^{|C|} \times \mathbb{R}_{>0} \times L \times \mathbb{N}^{|V|} \to \mathbb{R}_{\geqslant 0}^{|C|}$ увеличивает значения активных в данном состоянии таймеров на заданную вещественную величину:
   $inc((c_1,...,c_{|C|}),y,l,\overline{v}) = (c'_1,...,c'_{|C|})$, где

$$c'_i = \begin{cases} c_i + y, \text{ если } \overline{v} \in P(l,x_i), \text{ где } x_i \in C \text{ — } i\text{-ый таймер} \\ c_i, \text{ иначе} \end{cases}$$

На рисунке 2.1 приведен пример автомата, имеющего состояние $(l = A, c = 1, v = 2)$, для которого (состояния) выполнено условие $v = 2$ и, следовательно, определен дискретный переход $(l = A,\ c = 1,\ v = 2) \mapsto (l = B,\ c = 1,\ v = 0)$, а также непрерывные переходы $(l = A,\ c = 1,\ v = 2) \mapsto (l = A,\ c = 1 + d,\ v = 2)$, где $d \in (0,2)$. Для $d \geqslant 2$ соответствующих непрерывных переходов не существует, так как при таких значениях d инвариант $c < 3$, приписанный локации A, не выполняется.

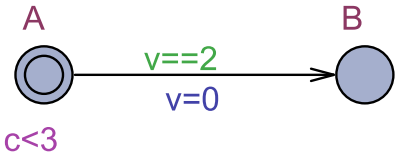

Рисунок 2.1 — Пример автомата с заданным состоянием

Несколько автоматов могут быть объединены в сеть автоматов. Обозначим за $A = A_1|...|A_n$ сеть временных автоматов, состоящую из автоматов $A_1,...A_n$, $A_i = \langle L_i, l_i^0, C, V, \overline{v_0}, AA, AS, E_i, I_i, P_i, U_i', U_i''\rangle$. Наборы таймеров ($C$) и переменных ($V$), начальные значения переменных ($\overline{v_0}$), множества действий (AS и AA) для всех автоматов сети общие. Локацией сети является совокупность локаций всех ее автоматов: $\bar{l} = (l_1,...,l_n)$. $\overline{l_0} = (l_1^0,...,l_n^0)$ — начальная локация. Обозначим как $\overline{aa} = (aa_1,...,aa_n)$ последовательность действий по изменению значений переменных и обнулению таймеров, выполняемых в автоматах сети как единое действие ($aa_i \in F(AA)$); aa_i может быть равно в т.ч. ε. Условия, при которых наборы действий из $F(AA)$ могут выполнятся как единое действие, а также порядок их выполнения будут приведены ниже, при описании возможных переходов (п.п. 2,3 стр. 44). $\overline{as} = (as_1,...,as_n)$ — действия синхронизации, выполняемые согласованно в автоматах сети ($as_i \in AS$); as_i может быть равно в т.ч. τ. Условия, при которых действия из AS могут выполнятся согласованно, также будут приведены ниже (п.п. 2,3 стр. 44). $\bar{l}[l_i'/l_i]$ — локация сети, в которой i-й элемент, т.е. l_i, заменен на l_i'. $I(\bar{l}) = \bigwedge\limits_i I_i(l_i)$ — инвариант локации сети. В одном из автоматов сети может быть выполнен некоторый переход, только если после его выполнения инвариант локации сети будет истинным, то есть инварианты текущих локаций всех автоматов будут истинными. $P(\bar{l},x) = \bigwedge\limits_i P_i(l_i, x)$ — условие активности таймера x в локации $\bar{l}$ сети.

Взаимодействие автоматов осуществляется за счет использования общих переменных, а также за счет выполнения действий синхронизации по каналам.

Под каналом понимается двойка (⟨имя канала⟩, ⟨тип канала⟩), где ⟨имя канала⟩ — произвольная строка, уникальная в рамках сети автоматов, ⟨тип канала⟩ — один из двух типов: «точка-точка», либо «широковещательный».

Под действием синхронизации понимается пометка дуги вида «<имя канала>?» или «<имя канала>!», обозначающая соответственно прием и отправку сигнала по каналу «точка-точка», либо по широковещательному каналу. Переходы по дугам, помеченным отправкой и приемом сигналов по одному и тому же каналу, выполняются одновременно. Разница между типами каналов заключается в том, что при взаимодействии по каналу «точка-точка» одновременно происходят ровно два перехода, помеченные соответственно как прием и отправка, а при взаимодействии по широковещательному каналу происходит ровно один переход, помеченный отправкой, и ноль или несколько переходов, помеченных приемом. Более формально взаимодействие по каналам будет описано ниже. Для каналов могут быть заданы приоритеты. Если для некоторого состояния сети автоматов в сети автоматов возможны синхронизации по нескольким различным каналам, то выполняется синхронизация по каналу с наивысшим приоритетом. Выбор канала среди каналов с одинаковыми приоритетами происходит недетерминированным образом.

Поведение сети задается системой переходов $\langle S, s_0, \mapsto\rangle$, где $S = (L_1 \times ... \times L_n) \times \mathbb{R}_{\geqslant 0}^{|C|} \times \mathbb{Z}^{|V|}$, $s_0 = (\overline{l_0}, \overline{c_0}, \overline{v_0})$ — начальное состояние, $\mapsto \subseteq ((S \times \mathbb{R}_{\geqslant 0} \times S) \cup (S \times (AS^n \times F(AA)^n) \times S)$ — отношение перехода. Аналогично переходам автоматов, переходы сети автоматов могут быть непрерывными и дискретными. Опишем, в каких случаях определены переходы каждого типа для сети автоматов:

1. определен дискретный переход $(\bar{l}, \bar{c}, \bar{v}) \xmapsto{\overline{as},\overline{aa}} (\bar{l}[l_i'/l_i], \overline{c'}, \overline{v'})$, где $as_m = \tau \quad \forall m \in \overline{1,n}$, $aa_m = aa_i$ при $m = i$, $aa_m = \varepsilon$ при $m \neq i$, если $\exists (l_i \xmapsto{b_i,\tau,aa_i} l_i') \in E_i : (\bar{c}, \bar{v}) \in b_i$, $(\overline{c'},\overline{v'}) \in I(\bar{l}[l_i'/l_i])$, а $(\overline{c'}, \overline{v'})$ получаются из $(\bar{c}, \bar{v})$ посредством выполнения действий aa_i.

То есть происходит дискретный переход в одном из автоматов сети. При этом новые значения таймеров и переменных должны удовлетворять инварианту новой локации сети.

2. определен дискретный переход $(\bar{l}, \bar{c}, \bar{v}) \xmapsto{\overline{as},\overline{aa}} (\bar{l}[l'_j/l_j, l'_i/l_i], \overline{c'}, \overline{v'})$, где

$$as_m = \begin{cases} as_j, \text{ при } m = j, \\ as_i, \text{ при } m = i, \\ \tau, \text{ в остальных случаях} \end{cases}$$

$$aa_m = \begin{cases} aa_j, \text{ при } m = j, \\ aa_i, \text{ при } m = i, \\ \varepsilon, \text{ в остальных случаях} \end{cases}$$

если $i \neq j$, $\exists(l_i \xmapsto{b_i,as_i,aa_i} l'_i) \in E_i$ и $\exists(l_j \xmapsto{b_j,as_j,aa_j} l'_j) \in E_j$, такие что $(\bar{c}, \bar{v}) \in (b_i \wedge b_j)$, $(\overline{c'},\overline{v'}) \in I(\bar{l}[l'_j/l_j, l'_i/l_i])$, а as_i и as_j — прием и отправка сигнала по каналу «точка-точка» с некоторым именем, а $(\overline{c'}, \overline{v'})$ получаются из $(\bar{c}, \bar{v})$ посредством выполнения действий aa_i, aa_j (при этом, если as_i — отправка сигнала, то сначала выполняются действия aa_i, иначе сначала выполняются действия aa_j). При таком переходе в сети происходит одновременная смена локаций двух ее автоматов.

3. определен дискретный переход $(\bar{l}, \bar{c}, \bar{v}) \xmapsto{\overline{as},\overline{aa}} (\bar{l}[l'_j/l_j, l'_{i_1}/l_{i_1}, ..., l'_{i_k}/l_{i_k}], \overline{c'}, \overline{v'})$, где

$$as_m = \begin{cases} as_j, \text{ при } m = j, \\ as_{i_h}, \text{ при } m = i_h, h \in \overline{1,k} \\ \tau, \text{ в остальных случаях} \end{cases}$$

$$aa_m = \begin{cases} aa_j, \text{ при } m = j, \\ aa_{i_h}, \text{ при } m = i_h, h \in \overline{1,k} \\ \tau, \text{ в остальных случаях} \end{cases}$$

если $j \neq i_h \forall h \in \overline{1,k}$, $i_{h_1} \neq i_{h_2} \forall h_1, h_2 \in \overline{1,k}$, $\exists(l_j \xmapsto{b_j,as_j,aa_j} l'_j) \in E_j$ и $\exists(l_{i_1} \xmapsto{b_{i_1},as_{i_1},aa_{i_1}} l'_{i_1}) \in E_{i_1}, ...$, $\exists(l_{i_k} \xmapsto{b_{i_k},as_{i_k},aa_{i_k}} l'_{i_k}) \in E_{i_k}$, такие что $(\bar{c}, \bar{v}) \in (b_j \wedge b_{i_1} \wedge ... \wedge b_{i_k})$, $(\overline{c'},\overline{v'}) \in I(\bar{l}[l'_j/l_j, l'_{i_1}/l_{i_1}, ..., l'_{i_k}/l_{i_k}])$, as_{i_1}, ..., as_{i_k}— прием сигнала по широковещательному каналу с заданным именем, as_j — отправка сигнала по широковещательному каналу с тем же именем, а $(\overline{c'}, \overline{v'})$ получаются из $(\bar{c}, \bar{v})$ посредством выполнения действий $aa_j, aa_{i_1}, ..., aa_{i_k}$ (при этом сначала выполняются действия aa_j, а затем $aa_{i_1}, ..., aa_{i_k}$ в порядке возрастания их индексов). При таком переходе в сети происходит одновременная смена локаций $(k+1)$ ее автоматов, k может быть равно 0 (происходит только отправка).

4. для некоторого $d \in \mathbb{R}_{\geqslant 0}$ определен непрерывный переход $(\bar{l}, \bar{c}, \bar{v}) \xmapsto{\overline{as},\overline{aa}} (\bar{l}, inc(\bar{c},d,\bar{l},\bar{v})), \bar{v})$, где $as_m = \tau$, $aa_m = \varepsilon \forall m \in \overline{1,n}$, если выполняется каждое из следующих условий:

   а) $\forall i : l_i \notin U'_i$;

   б) $\forall d' \in \mathbb{R}_{\geqslant 0} : d' \leqslant d$ справедливо соотношение $(inc(\bar{c},d',\bar{l},\bar{v}), \bar{v}) \in I(\bar{l})$.

   То есть локация сети автоматов остается прежней, и происходит увеличение значений таймеров, при котором инвариант локации должен оставаться истинным.

Функция $inc : \mathbb{R}_{\geqslant 0}^{|C|} \times \mathbb{R} \times L^n \times \mathbb{N}^{|V|} \to \mathbb{R}_{\geqslant 0}^{|C|}$ увеличивает значения всех активных в данной локации сети таймеров на заданную вещественную величину:

$inc((c_1, ..., c_{|C|}),y,\bar{l},\overline{v}) = (c'_1, ..., c'_{|C|})$, где

$$c'_i = \begin{cases} c_i + y, \text{ если } \overline{v} \in P(\bar{l},x_i), \text{ где } x_i \in C \text{ — } i\text{-ый таймер} \\ c_i, \text{ иначе} \end{cases}$$

Вычислением сети автоматов является всякая возможная конечная или бесконечная последовательность ее состояний вида $s_0 \mapsto s_1 \mapsto ... \mapsto s_n \mapsto ....$ *Поведение сети автоматов* — множество всех ее вычислений.

2.3. Обобщенные сети временных автоматов с остановкой таймеров

Для построения предлагаемой обобщенной модели функционирования МВС, абстрагированной от используемых в МВС алгоритмов планирования, и доказательства корректности этой модели автором разработан новый уровень абстракции временных автоматов с остановкой таймеров и их сетей — *обобщенные сети временных автоматов с остановкой таймеров*. Это расширение математического аппарата сетей временных автоматов позволяет абстрагироваться от систем переходов автоматов сети. Ниже приведен ряд определений, необходимых для введения определения обобщенной сети автоматов, и для описания обобщенной модели функционирования МВС.

Модельное время сети автоматов — значение некоторого служебного таймера, условие активности которого всегда истинно и который никогда не обнуляется. В терминах раздела 2.2 для сети автоматов $A = A_1|...|A_n$, состоящей из автоматов A_1,...A_n, $A_i = \langle L_i, l_i^0, C, V, \overline{v_0}, AA, AS, E_i, I_i, P_i, U'_i, U''_i \rangle$, это таймер $x_m \in C$, такой что, $\forall \bar{l} \ \forall \overline{v}$ верно, что $\overline{v} \in P(\bar{l},x_m)$ и «$x_m := 0$» $\notin AA$.

Событие синхронизации автоматов — это тройка $e = \langle CH, A^s, t \rangle$, соответствующая переходу вида 2 или 3 в сети автоматов (см. описание возможных видов переходов в разделе 2.2), где CH — канал синхронизации, $A^s \subseteq A$ — набор автоматов, участвующих в синхронизации, t — модельное время сети автоматов.

Переходу $(\bar{l}, \bar{c}, \overline{v}) \xmapsto{\overline{as},\overline{aa}} (\bar{l}[l'_j/l_j, l'_i/l_i], \overline{c'}, \overline{v'})$ (вида 2) соответствует событие синхронизации $\langle CH(as_j), \{A_i, A_j\}, c'_m \rangle$, где $CH(as_j) = CH(as_i)$ — имя канала, по которому происходит отправка и прием сигнала в действиях синхронизации as_j и as_i, A_j и A_i — автоматы, в которых выполняется отправка и прием сигнала, $c'_m = c_m$ — значение модельного времени.

Переходу $(\bar{l}, \bar{c}, \overline{v}) \xmapsto{\overline{as},\overline{aa}} (\bar{l}[l'_j/l_j, l'_{i_1}/l_{i_1}, ..., l'_{i_k}/l_{i_k}], \overline{c'}, \overline{v'})$ (вида 3) соответствует событие синхронизации $\langle CH(as_j), \{A_j, A_{i_1}, ..., A_{i_k}\}, c'_m \rangle$, где $CH(as_j) = CH(as_{i_1}) = ... = CH(as_{i_k})$ — имя широковещательного канала, по которому происходит отправка и прием сигнала в действиях синхронизации $as_j, as_{i_1}, ..., as_{i_k}$, $A_j, A_{i_1}, ..., A_{i_k}$ — автоматы, в которых выполняется отправка и прием сигнала, $c'_m = c_m$ — значение модельного времени.

Два события синхронизации $e = \langle CH, A^s, t\rangle$ и $\hat{e} = \langle \hat{CH}, \hat{A}^s, \hat{t}\rangle$ *совпадают*, если $CH = \hat{CH}$, $A^s = \hat{A}^s$ и $t = \hat{t}$.

Временная диаграмма (ВД) сети автоматов — набор событий синхронизации, соответствующий некоторому вычислению данной сети. Для того, чтобы по заданному вычислению сети автоматов получить ВД сети автоматов, необходимо выбрать из этого вычисления все переходы видов 2 и 3 и каждому переходу поставить в соответствие событие синхронизации.

Две ВД сети автоматов *эквивалентны*, если между их событиями можно установить взаимно однозначное соответствие, причем соответствующие друг другу события совпадают.

Согласно разделу 2.2, взаимодействие автоматов осуществляется посредством использования общих переменных и синхронизации по каналам. Набор переменных и действий синхронизации, посредством которых данный автомат взаимодействует с другими автоматами сети, называется *интерфейсом автомата*. Переменные, принадлежащие интерфейсу автомата, называются *переменными интерфейса*. Модификация и чтение значений этих переменных могут выполняться в различных автоматах сети. Переменные, не использующиеся для взаимодействия автоматов, т.е. переменные, модификация и чтение которых выполняются только в одном автомате сети, называются *внутренними переменными* этого автомата. В терминах раздела 2.2 интерфейсом автомата $A = \langle L, l_0, C, V, \overline{v_0}, AA, AS, E, I, P, U', U''\rangle$ является пара $\langle V^*, AS\rangle$, где $V^* \subseteq V$. Например, если некоторый автомат имеет интерфейс $\langle\{x,y\}, \{ch1?, ch1!, ch2!\}\rangle$, то это означает, что данный автомат взаимодействует с другими автоматами сети посредством модификации и чтения значений переменных x и y, приема сигналов по каналу $ch1$ и отправки сигналов по каналам $ch1$ и $ch2$.

Параметризованный автомат — это автомат, в арифметических и логических выражениях которого (то есть в предусловиях и действиях переходов, инвариантах локаций, условиях активности таймеров) наряду с числами и переменными используются целочисленные параметры с не заданными значениями. Формально, параметризованный автомат — это кортеж $A = \langle L, l_0, C, V, \overline{v_0}, AA, AS, E, I, P, U', U'', PM\rangle$, где $PM = (p_1, ..., p_{|PM|})$ — набор целочисленных параметров, а все остальные компоненты кортежа аналогичны соответствующим компонентам кортежа, определяющего автомат, описанный в разделе 2.2, с поправкой на использование параметров в AA, AS, I, P. Далее при описании модели МВС автомат, введенный в разделе 2.2 (непараметризованный автомат), будем называть *экземпляром автомата*. Экземпляр автомата порождается по параметризованному автомату путем задания константных числовых значений для параметров этого параметризованного автомата. Для непараметризованного автомата, как частного случая параметризованного, его экземпляр совпадает с ним самим.

Базовый тип автоматов определяется только интерфейсом и параметрами, то есть тройкой $\langle V, AS, PM\rangle$. Таким образом, использование базовых типов автоматов позволяет абстрагироваться от систем переходов автоматов.

Параметризованный автомат *реализует* базовый тип автоматов, если интерфейс параметризованного автомата совпадает с интерфейсом базового типа автоматов, и параметры параметризованного автомата совпадают с параметрами базового типа автоматов: $A = \langle L, l_0, C, V, \overline{v_0}, AA, AS, E, I, P, U', U'', PM\rangle$ реализует $A_b = \langle V_b, AS_b, PM_b\rangle$, если $V_b = V$, $AS_b = AS$, $PM_b = PM$.

Введенные уровни абстракции автоматов соответствуют в рамках предлагаемой схемы моделирования уровням абстракции компонентов МВС (см. таблицу 2.2). Базовому типу автоматов соответствует базовый тип компонентов. Базовый тип компонента МВС определяет сервис, предоставляемый компонентом системы, но не логику функционирования этого компонента. Примером базового типа компонента МВС является планировщик раздела. Параметризованному автомату соответствует конкретный тип компонента МВС. Конкретный тип компонента МВС определяет детали реализации базового типа компонента МВС. Примером конкретного типа компонента МВС является планировщик раздела, работающий по алгоритму планирования с фиксированными приоритетами и вытеснением (FPPS). Экземпляру автомата соответствует экземпляр компонента МВС.

Таблица 2.2 — Уровни абстракции компонентов МВС и моделирующих их автоматов

Уровень абстракции компонента системы	*Уровень абстракции автомата*	*Пример компонента*
Базовый тип компонента	Базовый тип автомата	Планировщик раздела
Конкретный тип компонента	Параметризованный автомат	Планировщик, работающий по стратегии FPPS
Экземпляр компонента	Экземпляр автомата	Планировщик раздела <<partition10>>

Набор базовых типов автоматов назовем *обобщенной сетью автоматов*. Набор параметризованных автоматов — *параметризованной сетью автоматов*, сеть автоматов, введенную в разделе 2.2, — *экземпляром сети автоматов*.

2.4. Описание обобщенной модели функционирования МВС

Автором предложено представление обобщенной модели функционирования МВС в виде обобщенной сети автоматов с остановкой таймеров.

В предлагаемой обобщенной сети имеются следующие средства взаимодействия автоматов (обозначения M, K_i, H, T_{ij}, pr_{ij} были введены в разделе 1.4.1 и соответствуют характеристикам конфигурации МВС):

- переменные $\texttt{is\_ready}_{ij}$, $\texttt{abs\_deadline}_{ij}$, $i \in \overline{1,M}, j \in \overline{1,K_i}$, $\texttt{is\_data\_ready}_h$, $h \in \overline{1,H}$, соответствующие готовности очередной работы задачи T_{ij}, правой границе директивного интервала этой работы и доставке h-го сообщения;
- переменные $\texttt{prio}_{ij}$, $\texttt{id}_{ij}$, $i \in \overline{1,M}, j \in \overline{1,K_i}$, значения каждой из которых равны приоритету (pr_{ij}) и уникальному в рамках раздела идентификатору(j) задачи T_{ij}; значения этих переменных не изменяются в процессе функционирования сети;
- каналы $\texttt{wakeup}_i$, $\texttt{sleep}_i$, $i \in \overline{1,M}$, синхронизации по которым соответствуют началу и завершению очередного окна i-го раздела;

– каналы $\texttt{ready}_i$, $\texttt{finished}_i$, $i \in \overline{1,M}$, синхронизации по которым соответствуют оповещению планировщика раздела о готовности и завершении (вследствие окончания выполнения, либо достижения правой границы директивного интервала) некоторой работы i-го раздела;
– каналы $\texttt{exec}_{ij}, \texttt{preempt}_{ij}, i \in \overline{1,M}, j \in \overline{1,K_i}$, синхронизации по которым соответствуют постановке на выполнение и вытеснению очередной работы задачи T_{ij};
– широковещательные каналы $\texttt{send}_{ij}$, $\texttt{receive}_{ij}$, $i \in \overline{1,M}$, $j \in \overline{1,K_i}$, синхронизации по которым соответствуют отправке и приему сообщений очередной работой задачи T_{ij}.

Приоритеты назначены перечисленным каналам так, что в соответствии с этими приоритетами каналы частично упорядочены следующим образом:

$\texttt{exec}_{ij} = \texttt{send}_{ij} = \texttt{receive}_{ij} < \texttt{ready}_i < \texttt{sleep}_i = \texttt{wakeup}_i < \texttt{preempt}_{ij} < \texttt{finished}_i$

Обобщенная сеть автоматов, моделирующая функционирование МВС, состоит из следующих базовых типов автоматов.

1. Базовый тип автоматов **T**, моделирующий функциональную задачу и определенный следующим интерфейсом (i — номер раздела, которому принадлежит задача, j — номер задачи в разделе):

– получение сигналов по каналам $\texttt{exec}_{ij}$ и $\texttt{preempt}_{ij}$ и отправка сигналов по каналам $\texttt{ready}_i$ и $\texttt{finished}_i$;
– отправка сигналов по широковещательному каналу $\texttt{send}_{ij}$ и получение сигналов по широковещательному каналу $\texttt{receive}_{ij}$;
– переменные $\texttt{is\_ready}_{ij}$ и $\texttt{is\_data\_ready}_h$, где h — номер сообщения, которое получают работы функциональной задачи (для одной задачи может быть определено несколько сообщений);
– переменные $\texttt{abs\_deadline}_{ij}$.

Базовый тип автоматов **T** имеет следующие целочисленные параметры, соответствующие характеристикам функциональной задачи (i — номер раздела, которому принадлежит задача, j — номер задачи в разделе; для удобства параметры сгруппированы в структуру `data`):

– `data.wcet` — длительность выполнения в худшем случае (WCET) на соответствующем ядре $c_{ij}^{CType(Bind(Part_i))}$;
– `data.period` — период (p_{ij});
– `data.offset` — смещение относительно начала периода, определяющее левые границы директивных интервалов работ (o_{ij});
– `data.deadline` — смещение относительно начала периода, определяющее правые границы директивных интервалов работ (d_{ij});
– `data.numOfInputLinks` — количество входных сообщений для задачи (мощность множества $\{Msg_h = \langle S_h, R_h, dm_h, dn_h\rangle \in Msg | S_h = T_{ij}\}$).

2. Базовый тип автоматов **TS**, моделирующий планировщик работ раздела и определенный следующим интерфейсом (i — номер раздела):

– получение сигналов по каналам $\texttt{wakeup}_i$, $\texttt{sleep}_i$, $\texttt{ready}_i$ и $\texttt{finished}_i$;
– отправка сигналов по каналам $\texttt{exec}_{ij}$, $\texttt{preempt}_{ij}$, где j-й канал соответствует j-й задаче раздела;

– переменные $\texttt{is\_ready}_{ij}$, $\texttt{prio}_{ij}$, $\texttt{id}_{ij}$, $\texttt{abs\_deadline}_{ij}$ (значения перечисленных переменных могут потребоваться для выбора работы, которая должна быть поставлена на выполнение в некоторый момент времени; модель планировщика работ раздела может лишь читать значения этих переменных).

Базовый тип автоматов **TS** имеет один целочисленный параметр `numOfTasks` — количество задач в i-м разделе (K_i).

3. Базовый тип автоматов **CS**, моделирующий планировщик вычислительного ядра. Планировщик ядра переключает окна разделов на конкретном ядре согласно статическому расписанию, являющемуся элементом конфигурации МВС. **CS** определен следующим интерфейсом:

– отправка сигналов по каналам $\texttt{wakeup}_i$ и $\texttt{sleep}_i$, где i-й канал соответствует i-му разделу.

Базовый тип автоматов **CS** имеет целочисленный параметр `majorFrame`, задающий длительность интервала планирования (L) и массивы параметров `windows.start`, `windows.stop`, `windows.partitionId`, соответствующие расписанию окон $Sched$ для вычислительного ядра; размер массивов задается параметром `numOfWindows`.

4. Базовый тип автоматов **L**, моделирующий виртуальный канал. Под виртуальным каналом в данном случае понимается не виртуальный канал AFDX [43] или FC-AE-ASM-RT [44], а средство передачи некоторого сообщения в общем случае. Это соответствует определению канала в стандарте ARINC 653 [45]. Одному сообщению взаимно однозначно соответствует один виртуальный канал. **L** определен следующим интерфейсом:

– получение сигналов по широковещательному каналу $\texttt{send}_{i_1 j_1}$ и отправка сигналов по широковещательному каналу $\texttt{receive}_{i_2 j_2}$, где i_1, j_1, i_2, j_2 таковы, что для сообщения, которому соответствует моделируемый виртуальный канал, задача $T_{i_1 j_1}$ является отправителем, а задача $T_{i_2 j_2}$ — получателем;
– изменение переменной $\texttt{is\_data\_ready}_h$, где h — номер сообщения, передаваемого по моделируемому виртуальному каналу.

Базовый тип автоматов **L** имеет один целочисленный параметр `delay` — длительность передачи сообщения по виртуальному каналу (dm_h или dn_h — в зависимости от расположения задачи-отправителя и задачи-получателя к одному или разным модулям). Как правило, в системах ИМА через конкретные виртуальные каналы передаются однотипные сообщения фиксированного размера, поэтому для модели виртуального канала достаточно одного параметра, задающего задержку.

Средства взаимодействия описанных базовых типов автоматов показаны на рисунке 2.2. Синхронизации по каналам обозначены стрелками; направление стрелки соответствует направлению передачи сигнала — от отправителя к получателю. Переменные интерфейса обозначены пунктирными линиями; пометка «r» на конце такой линии обозначает, что переменная доступна автомату на чтение, «w» – на запись, «r/w» — на чтение и запись.

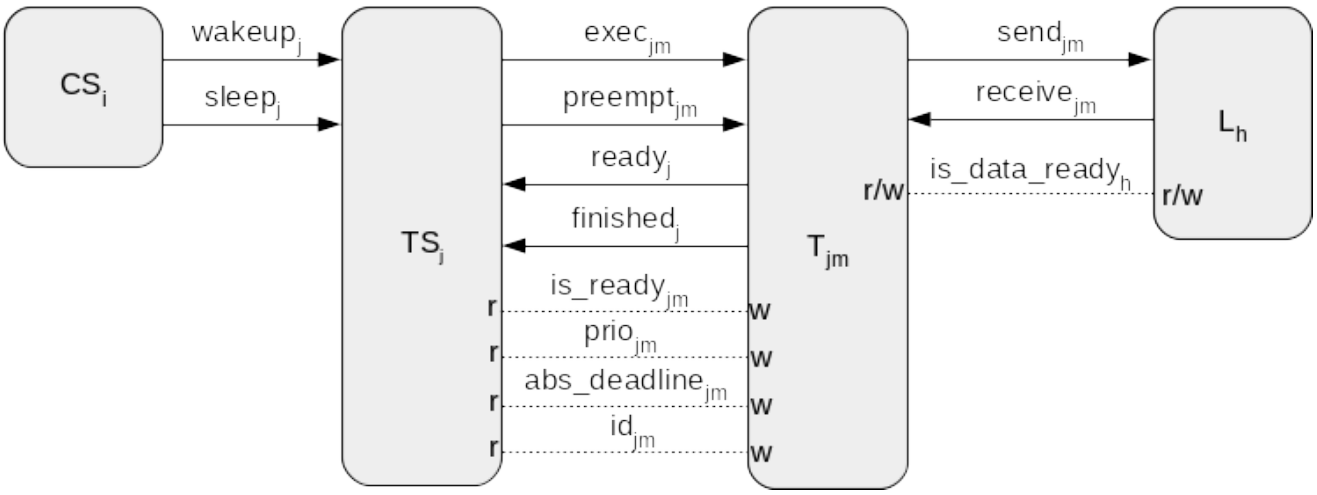

Рисунок 2.2 — Средства взаимодействия базовых типов автоматов в обобщенной модели функционирования МВС

Поскольку обобщенная сеть автоматов абстрагирована от систем переходов автоматов, а логика функционирования компонентов МВС (в т.ч. алгоритмы планирования) описывается при помощи систем переходов соответствующих автоматов-моделей, то предложенная обобщенная модель функционирования МВС является абстрагированной от используемых в МВС алгоритмов планирования. Обобщенная модель функционирования МВС не описывает конкретные числовые характеристики и количество компонентов МВС, поэтому она также является абстрагированной от структуры МВС.

2.5. Метод построения экземпляра модели по описанию конфигурации МВС

Параметризованная модель МВС — это параметризованная сеть автоматов, состоящая из параметризованных автоматов, которые реализуют базовые типы автоматов, формирующие предложенную в разделе 2.4 обобщенную модель функционирования МВС (т.е. обобщенную сеть автоматов). Параметрами параметризованной модели являются числа, которые задают выбор составляющих конфигурации МВС и характеристики этих составляющих.

Параметризованная модель МВС содержит следующие разработанные автором параметризованные автоматы: модели задачи, планировщика ядра, виртуального канала и нескольких планировщиков работ, работающих согласно различным алгоритмам планирования (с фиксированными и динамическими приоритетами, с вытеснением и без вытеснения). Описание этих параметризованных автоматов приведено в разделе 2.7 и приложении А. Разработанная параметризованная модель может быть дополнена новыми параметризованными автоматами, реализующими описанные в разделе 2.4 базовые типы автоматов.

Экземпляр этой параметризованной сети автоматов является моделью МВС с фиксированной конфигурацией.

Структура модели МВС с фиксированной конфигурацией показана на рисунке 2.3. Стрелки соответствуют направлениям передачи сигналов. Так в экземпляре автомата, реализующем базовой тип автоматов **CS** (т.е. в модели планировщика ядра), осуществляется посылка сигналов, соответствующих открытию и закрытию окон разделов. Прием этих сигналов осуществляется в экземпляре автомата, реализующем базовой тип автоматов **TS** (т.е. в модели планировщика ядра). Передача сигналов в обратном направлении (от **TS** к **CS**) не предполагается. В то же время между экземплярами автоматов, реализующими базовые типы автоматов **TS** и **T**, взаимодействие осуществляется в двух направлениях: модель планировщика раздела посылает сигналы, соответствующие постановке на выполнение и вытеснению работ, а модель функциональной задачи — соответствующие готовности и завершению работ. Направление передачи сигналов между экземплярами автоматов, реализующими базовые типы автоматов **T** и **L**, зависит от того, моделирует ли экземпляр автомата, реализующий базовый тип автоматов **T**, задачу-отправителя сообщения или задачу-получателя сообщения: передача сигналов осуществляется от модели задачи-отправителя к модели виртуального канала и от модели виртуального канала к модели задачи-получателя.

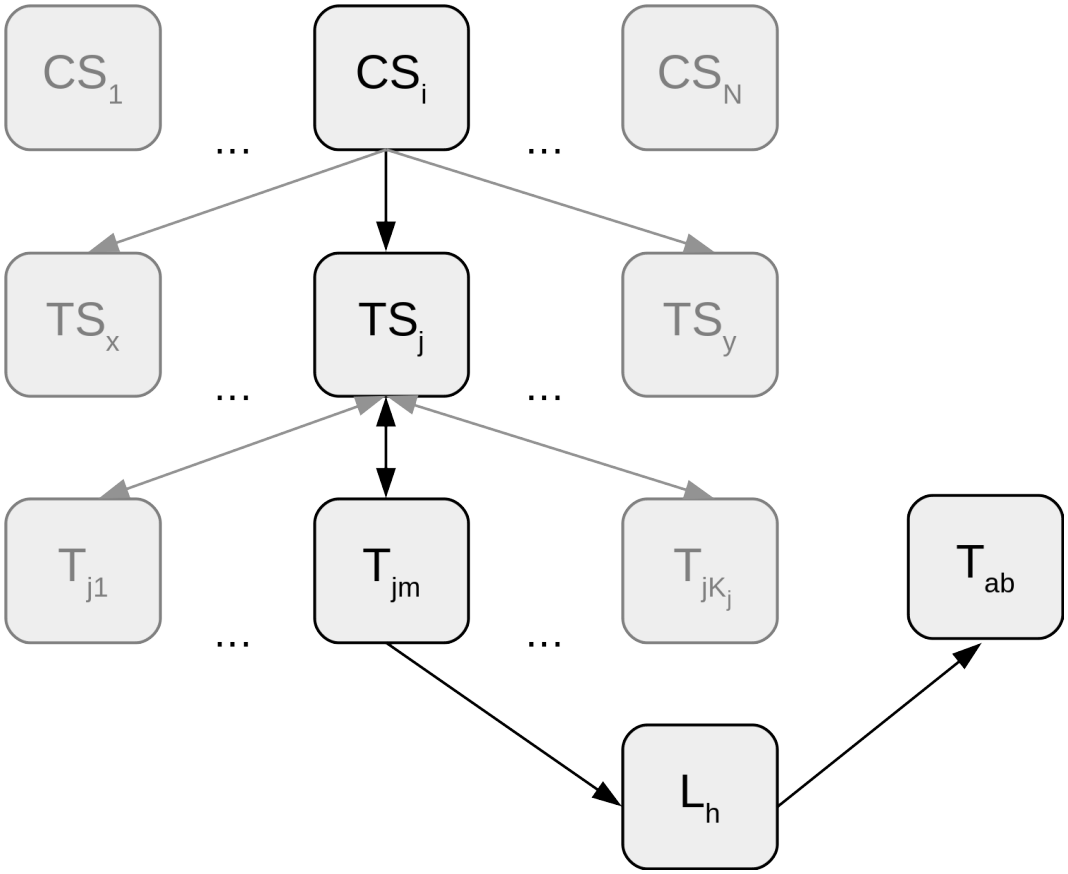

Рисунок 2.3 — Структура модели МВС с фиксированной конфигурацией

Для разработанной параметризованной сети автоматов, реализующей предложенную обобщенную модель функционирования МВС, и конфигурации МВС $conf \in CONF$ (обозначения составляющих конфигурации см. в разделе 1.4.1) экземпляр сети автоматов, моделирующий МВС с этой конфигурацией, строится согласно следующему алгоритму:

- цикл по ядрам ($i \in \overline{1, N}$):
  - ◇ цикл по разделам ядра ($j \in \overline{1, M} : Bind(Part_j) = HW_i$):
    - ∗ создание каналов $\texttt{ready}_j$, $\texttt{finished}_j$, $\texttt{wakeup}_j$, $\texttt{sleep}_j$;
    - ∗ цикл по задачам раздела ($k \in \overline{1, K_j}$):

      — создание каналов $\texttt{exec}_{jk}$, $\texttt{preempt}_{jk}$, $\texttt{send}_{jk}$, $\texttt{receive}_{jk}$;

      — создание переменных $\texttt{is\_ready}_{jk}$, $\texttt{abs\_deadline}_{jk}$, $\texttt{prio}_{jk}$ и инициализация переменной $\texttt{prio}_{jk}$ соответствующим значением из $conf$;

      — создание переменных $\texttt{is\_data\_ready}_h$, соответствующих виртуальным каналам, для которых задача T_{jk} является получателем, и инициализация их значением 0;

      — порождение параметризованного автомата, реализующего базовый тип **T** (моделирующего функциональную задачу), отождествление каналов и переменных его интерфейса с каналами $\texttt{exec}_{jk}$, $\texttt{preempt}_{jk}$, $\texttt{send}_{jk}$, $\texttt{receive}_{jk}$, $\texttt{ready}_j$, $\texttt{finished}_j$ и переменными $\texttt{is\_ready}_{jk}$, $\texttt{prio}_{jk}$, $\texttt{abs\_deadline}_{jk}$, $\texttt{is\_data\_ready}_h$, и установка его параметров в соответствующие значения из $conf$;
    - ∗ порождение параметризованного автомата, реализующего базовый тип **TS** (моделирующего планировщик работ) и соответствующего алгоритму планирования a_j j-го раздела, отождествление каналов и переменных его интерфейса с каналами $\texttt{exec}_{jk}$, $\texttt{preempt}_{jk}$, $\texttt{ready}_j$, $\texttt{finished}_j$, $\texttt{wakeup}_j$, $\texttt{sleep}_j$ и переменными $\texttt{is\_ready}_{jk}$, $\texttt{prio}_{jk}$, $\texttt{abs\_deadline}_{jk}$ ($k \in \overline{1, K_j}$);
  - ◇ порождение параметризованного автомата, реализующего базовый тип **CS** (моделирующего планировщик ядра), отождествление каналов его интерфейса с каналами $\texttt{wakeup}_j$, $\texttt{sleep}_j$ ($j \in \overline{1, M} : Bind(Part_j) = HW_i$), и установка его параметров в соответствующие i-му ядру значения из $Sched$;
- цикл по виртуальным каналам ($h \in \overline{1, H}$):
  - ◇ порождение параметризованного автомата, реализующего базовый тип **L** (моделирующего виртуальный канал), отождествление каналов и переменной его интерфейса с соответствующими каналами $\texttt{send}_{j_1k_1}$, $\texttt{receive}_{j_2k_2}$ и переменной $\texttt{is\_data\_ready}_h$, и установка его параметров в соответствующие значения из $conf$ (j_1, k_1, j_2, k_2 таковы, что для h-го сообщения задача $T_{j_1k_1}$ является отправителем, а задача $T_{j_2k_2}$ — получателем);

После установки значений параметров параметризованного автомата в конкретные числа из описания конфигурации МВС, этот параметризованный автомат становится экземпляром автомата. Сеть из этих экземпляров автоматов и является моделью МВС с заданной конфигурацией. Таким образом, предложенный алгоритм позволяет конкретизировать обобщенную модель на случай МВС с заданной конфигурацией.

Согласно предложенному алгоритму, каждому компоненту МВС, описанному в конфигурации (функциональной задаче, разделу, вычислительному ядру, сообщению), однозначно

соответствует экземпляр автомата (модель функциональной задачи, планировщика работ раздела, планировщика ядра, виртуального канала). Параметры автоматов однозначно устанавливаются в соответствующие значения из описания конфигурации. Связи между экземплярами автоматов также определяются однозначно по описанию конфигурации:

- $\forall i \in \overline{1, N}$ модель планировщика ядра HW_i взаимодействует только с моделями планировщиков разделов $Part_j$, $j \in \overline{1, M}$, таких что $Bind(Part_j) = HW_i$;
- $\forall j \in \overline{1, M}$ модель планировщика работ раздела $Part_j$ взаимодействует только с моделями функциональных задач T_{jk}, $k \in \overline{1, K_i}$ этого раздела и моделью планировщика ядра HW_i, $i \in \overline{1, N}$, такого что $Bind(Part_j) = HW_i$;
- $\forall j \in \overline{1, M}, \forall k \in \overline{1, K_j}$ модель функциональной задачи T_{jk} взаимодействует только с моделью планировщика раздела $Part_j$ и моделями виртуальных каналов, осуществляющих передачу сообщений Msg_h, $h \in \overline{1, H}$ ($Msg_h = \langle S_h, R_h, dm_h, dn_h \rangle$, см. раздел 1.4.1), таких что $S_h = T_{jk}$ (T_{jk} является отправителем h-го сообщения; в этом случае взаимодействие осуществляется посредством канала $\texttt{send}_{jk}$) или $R_h = T_{jk}$ (T_{jk} является получателем h-го сообщения; в этом случае взаимодействие осуществляется посредством канала $\texttt{receive}_{jk}$);
- $\forall h \in \overline{1, H}$ модель виртуального канала, осуществляющего передачу сообщения Msg_h, взаимодействует только с моделями функциональных задач T_{jk}, $j \in \overline{1, M}, k \in \overline{1, K_j}$, таких что $S_h = T_{jk}$ или $R_h = T_{jk}$.

Таким образом, каждой конфигурации МВС *однозначно* соответствует экземпляр модели МВС (по построению).

2.6. Построение временной диаграммы функционирования МВС

Каждому вычислению экземпляра сети автоматов соответствует ВД этой сети автоматов. Метод построения ВД сети автоматов по ее вычислению приведен в разделе 2.3. Далее в разделе 3.4 будет доказано, что в рамках предложенной обобщенной модели функционирования МВС для всех вычислений заданного экземпляра сети автоматов соответствующие ВД эквивалентны.

Чтобы по некоторой ВД сети автоматов, моделирующей МВС с заданной конфигурацией, построить временную диаграмму функционирования соответствующей МВС, использующуюся для проверки выполнения ограничений реального времени, необходимо выполнить следующие действия:

1. Исключить из ВД сети автоматов события $\langle CH, A^s, t \rangle$, такие что $CH \notin \{\texttt{exec}_{jk}, \texttt{preempt}_{jk}, \texttt{finished}_j\}$, $j \in \overline{1, M}, k \in \overline{1, K_j}$. Синхронизации по перечисленным каналам моделируют постановку на выполнение, вытеснение и завершение работ. Соответствующие события должны присутствовать в ВД МВС (согласно определению ВД МВС в разделе 1.4.2). Синхронизации по другим каналам моделируют события, временные метки которых напрямую не используются для проверки выполнения ограничений реального времени, поэтому такие события не включаются в ВД МВС.

2. Для каждого из оставшихся в ВД сети автоматов события $\langle CH, A^s, t\rangle$ поместить в ВД МВС (изначально пустую) событие $\langle EType, Src, x\rangle$, формируемое по следующим правилам:
   - тип события $EType$ однозначно определяется каналом синхронизации: $EType = \begin{cases} EX, \text{ если } CH = \texttt{exec}_{jk} \\ PR, \text{ если } CH = \texttt{preempt}_{jk} \\ FIN, \text{ если } CH = \texttt{finished}_j \end{cases}$
   - работа-источник события Src однозначно определяется автоматом-моделью функциональной задачи, участвующим в синхронизации:$Src = w_{ijk}$, где $\langle i, j\rangle$ — номер автомата $\mathtt{T}_{ij} \in A^s$, моделирующего задачу T_{ij}, $k = \lfloor t/p_{ij} \rfloor + 1$ (для EX) и $k = \lceil t/p_{ij} \rceil$ (для PR и FIN); таким образом, если время синхронизации равно $q \cdot p_{ij}$ (целому числу периодов), то событие типа EX относится к $(q+1)$-й работе, а событие типа PR или FIN — к q-й; по построению обобщенной модели функционирования МВС для любой синхронизации по каналам перечисленных в п. 1 типов одним из участников синхронизации является модель некоторой функциональной задачи, причем в каждой синхронизации такая модель ровно одна.
   - Время события в ВД МВС однозначно определяется временем соответствующей синхронизации в сети автоматов: $x = t$.

Согласно описанной процедуре, временной диаграмме сети автоматов, моделирующей МВС с заданной конфигурацией, *однозначно* (по построению) ставится в соответствие временная диаграмма функционирования МВС.

2.7. Разработанные модели компонентов МВС

Автором был разработан ряд параметризованных автоматов, которые реализуют базовые типы автоматов, описанные в разделе 2.4. Эти параметризованные автоматы представляют собой модели планировщика ядра, функциональной задачи и виртуального канала, а также модели трех часто используемых в МВС планировщиков работ раздела: с фиксированными приоритетами и вытеснением [45], с фиксированным приоритетами без вытеснения [47], планировщик, работающий по стратегии EDF с вытеснением [46]. Далее описана логика функционирования перечисленных моделей. Подробное описание соответствующих автоматов приведено в приложении А.

Во введенных выше обозначениях каналов ($\texttt{exec}_{ij}$, $\texttt{preempt}_{ij}$, $\texttt{send}_{ij}$, $\texttt{receive}_{ij}$, $\texttt{ready}_i$, $\texttt{finished}_i$, $\texttt{sleep}_i$, $\texttt{wakeup}_i$) и переменных ($\texttt{is\_ready}_{ij}$, $\texttt{prio}_{ij}$, $\texttt{abs\_deadline}_{ij}$) индекс i соответствует номеру раздела, а j — номеру задачи в разделе. Индекс h в обозначении $\texttt{is\_data\_ready}_h$ соответствует номеру виртуального канала. Так как экземпляр модели планировщика работ некоторого раздела по построению взаимодействует с моделями задач только

этого раздела, то индекс i в рамках модели планировщика работ раздела всегда один и тот же. Поэтому при описании моделей планировщиков работ раздела индекс, соответствующий номеру раздела, будет опущен. Аналогично, в рамках модели задачи можно опустить индексы i и j, а в рамках модели виртуального канала — индексы i, j, h.

2.7.1. Модель планировщика ядра

Модель циклически отрабатывает это расписание: в момент открытия очередного окна посылает модели планировщика i-го раздела сигнал по каналу $\texttt{wakeup}_i$, в момент закрытия окна — по каналу $\texttt{sleep}_i$ (где i — номер раздела, которому принадлежит данное окно).

2.7.2. Модели планировщиков работ разделов

Перед описанием моделей конкретных планировщиков отметим, что все они управляют постановкой работ на выполнение, возможно, с вытеснением других работ. При этом снятие опоздавших работ с вычислителя отрабатывается моделями функциональных задач, которые посылают модели планировщика сигнал по каналу `finished` как в случае штатного завершения работ, так и в случае опоздания работ.

1. **Планировщик с фиксированными приоритетами и вытеснением.**

Информацию о приоритетах задач соответствующего раздела модель получает посредством набора переменных $\texttt{prio}_i$, где i — номера задач раздела.

Пока модель не получит сигнал по своему каналу `wakeup`, она находится в неактивном состоянии. Как только модель получает сигнал по каналу `wakeup` (открытие окна данного раздела), она переходит в активное состояние. В активном состоянии модель находится до тех пор, пока не получит сигнал по каналу `sleep` (закрытие окна раздела).

При переходе модели планировщика в активное состояние происходит определение работы, которая должна быть поставлена на выполнение в данный момент. Определение выполняется на основе значений переменных $\texttt{is\_ready}_i$, каждая из которых является флагом готовности текущей работы i-й задачи раздела, и приоритетов: на выполнение должна быть поставлена та работа из набора готовых работ, которая имеет максимальный приоритет. Далее либо моделируется постановка соответствующей работы на выполнение (посылка сигнала по каналу $\texttt{exec}_i$, где i — номер задачи, которой принадлежит работа), либо (если ни одной готовой работы нет) происходит ожидание получения какого-либо сигнала.

При получении в активном состоянии сигнала по каналу `ready`, соответствующего оповещению о готовности работы некоторой задачи, снова происходит определение работы, которая должна быть поставлена на выполнение в данный момент. Если уже моделируется выполнение

работы с максимальным среди готовых работ приоритетом (новая готовая работа имеет меньший приоритет), то больше никаких действий в модели не происходит до следующего получения какого-либо сигнала. Если в данный момент моделируется выполнение работы с приоритетом, меньшим чем у новой готовой работы, то моделируется вытеснение выполняющейся работы (посылка сигнала по каналу $\texttt{preempt}_i$, где i — номер задачи, которой принадлежит выполняющаяся работа) и постановка на выполнение новой работы (посылка сигнала по каналу $\texttt{exec}_j$, где j — номер задачи, которой принадлежит новая работа). Если в данный момент выполняющихся работ нет, то сразу моделируется постановка новой работы на выполнение (посылка сигнала по каналу $\texttt{exec}_j$).

Если модель планировщика раздела находится в активном состоянии и несколько моделей функциональных задач готовы отправить сигналы по каналу `ready` модели планировщика раздела, то в модели планировщика раздела сначала произойдет получение всех сигналов по каналу `ready`, а лишь потом определение работы, которая должна быть поставлена на выполнение в данный момент. Порядок получения сигналов по каналу `ready` при этом на выбор работы для постановки на выполнение не влияет. Такое поведение моделей достигается за счет назначения приоритетов каналов, описанного в разделе 2.4: приоритет канала `ready` больше, чем приоритет каналов $\texttt{exec}_j$.

Если в момент, соответствующий выполнению некоторой работы, получен сигнал по каналу `finished` (штатное окончание выполнения, либо достижение правой границы директивного интервала выполняющейся работы), то вновь происходят действия по определению работы, которая должна быть поставлена на выполнение в данный момент, и в случае необходимости моделируется постановка на выполнение новой работы. Если в момент, соответствующий выполнению некоторой работы, получен сигнал по каналу `sleep`, то перед переходом модели в неактивное состояние моделируется вытеснение этой работы (посылка сигнала по каналу $\texttt{preempt}_i$).

2. **Планировщик, работающий по стратегии EDF с вытеснением.**

Данная модель аналогична модели планировщика с фиксированными приоритетами и вытеснением, но обладает следующими отличиями:

- определение работы, которая должна быть поставлена на выполнение в данный момент времени, осуществляется согласно стратегии EDF: на выполнение ставится та работа из множества готовых работ, правая граница директивного интервала которой является наименьшей; если таких готовых работ несколько, то выбирается работа с наименьшим номером.
- информацию о номере задачи и о значении правой границы директивного интервала для каждой работы модель получает посредством наборов переменных $\texttt{id}_i$ и $\texttt{abs\_deadline}_i$, где i — номера задач раздела; в каждый момент времени в МВС может существовать не более одной работы каждой задачи и переменная $\texttt{abs\_deadline}_i$ содержит значение правой границы директивного интервала текущей работы i-й задачи раздела; значения переменных $\texttt{abs\_deadline}_i$ вычисляются в моделях функциональных задач.

3. **Планировщик с фиксированными приоритетами без вытеснения.**

Данная модель аналогична модели планировщика с фиксированными приоритетами и вытеснением, но обладает следующим отличием: выполняющаяся работа может быть вытеснена (модель планировщика может послать сигнал по каналу $\texttt{preempt}_i$) только в случае закрытия текущего окна раздела, при этом именно вытесненная работа ставится на выполнение при открытии следующего окна этого раздела (если эта работа не становится опоздавшей к моменту открытия окна). Если в рамках окна соответствующего раздела выполняется некоторая работа, то новая работа того же раздела может быть поставлена на выполнение только после завершения уже выполняющейся работы, в т.ч. по причине опоздания (в связи с чем модель планировщика получит сигнал по каналу `finished`).

2.7.3. Модель функциональной задачи

Модель имеет несколько состояний, соответствующих состояниям работ функциональной задачи:

1. Ожидание наступления левой границы директивного интервала.
2. Ожидание получения сообщений от работ-отправителей. Переход в это состояние может произойти из состояния 1, если к моменту достижения левой границы директивного интервала очередная работа получила сообщения не от всех работ-отправителей. Получение сообщения от некоторой работы-отправителя моделируется с помощью получения сигнала по каналу `receive` и чтения переменных $\texttt{is\_data\_ready}_h$.
3. Ожидание постановки на выполнение. Переход в это состояние происходит из состояния 1 при достижения левой границы директивного интервала работы, если сообщения от всех работ-отправителей к этому моменту получены, либо эта задача не зависит по данным от других задач; из состояния 2 при получении сообщений от всех работ-отправителей или из состояния 4 при вытеснении выполняющейся работы (вытеснение моделируется с помощью получения сигнала по каналу `preempt`). При переходе в состояние 3 переменной `is_ready` присваивается значение 1.
4. Выполнение. Переход в это состояние происходит из состояния 3 при получении сигнала по каналу `exec` от модели планировщика. Для каждой работы суммарное время нахождения модели в этом состоянии должно быть равно заданной для задачи длительности выполнения на вычислительном ядре в худшем случае.

Также модель имеет состояние 5, соответствующее ожиданию наступления следующего периода, то есть отсутствию текущей работы и ожиданию появления новой работы. Переход в это состояние может произойти из состояний 2—4 и осуществляется либо после того, как промоделировано штатное завершение работы (посылка сигнала по каналу `finished`) с отправкой сообщений всем работам-получателям (посылка сигнала по широковещательному каналу `send`), либо после того, как промоделировано опоздание работы (для выполняющейся работы при этом тоже происходит посылка сигнала по каналу `finished`). При переходе в состояние

5 переменной `is_ready` присваивается значение 0. При наступлении следующего периода из этого состояния осуществляется переход в состояние 1, во время которого обновляется переменная `abs_deadline` (значение правой границы директивного интервала следующей работы устанавливается равным соответствующему значению для предыдущей работы, увеличенному на значение периода задачи). Для первой работы значение переменной `abs_deadline` равно указанному в конфигурации смещению правой границы директивного интервала задачи относительно начала периода.

2.7.4. Модель виртуального канала

Напомним, что под виртуальным каналом в данной работе понимается средство передачи сообщений между функциональными задачами, что соответствует интерпретации, принятой в стандарте ARINC 653 [45].

Модель получает сигнал по широковещательному каналу `send`, отрабатывает необходимую задержку и отправляет сигнал по каналу `receive`.

2.8. Оценки сложности построения и прогона экземпляра модели МВС

2.8.1. Сложность построения экземпляра МВС

Оценим сложность построения экземпляра модели МВС в зависимости от характеристик соответствующей конфигурации МВС. Построение выполняется согласно алгоритму, предложенному в разделе 2.5.

При порождении экземпляра автомата должны быть порождены объекты, соответствующие всем его локациям и переходам. Сложность порождения каждого объекта, соответствующего локации или переходу, константна. Количество локаций и переходов в одном автомате можно ограничить некоторыми константами. Так для всех разработанных автором параметризованных автоматов, описание которых приведено в разделе 2.7 и приложении А, количество локаций не превышает 10, а количество переходов не превышает 20. Таким образом, сложность порождения одного автомата можно считать константной. Сложность порождения одного канала и одной переменной константна, как и сложность отождествления одного канала с другим (одной переменный с другой). Сложность установки одного параметра автомата в соответствующее значение из описания конфигурации также константна.

Алгоритм, описанный в разделе 2.5, представляет собой два последовательно выполняемых цикла: цикл по ядрам и цикл по виртуальным каналам. Оценим сложность каждого из этих циклов.

Цикл по ядрам содержит вложенный цикл по привязанным к ядру разделам, который в свою очередь содержит вложенный цикл по задачам раздела.

На одной итерации цикла по задачам раздела порождается 1 автомат, 4 канала, (h_{ijk} + 3) переменных (h_{ijk} — количество входных сообщений у k-й задачи j-го раздела i-го ядра), происходит установка 4-х параметров в значения из описания конфигурации, выполняется 6 операций отождествления для каналов и 4 операции отождествления для переменных. Таким образом, сложность одной итерации цикла по задачам j-го раздела имеет порядок $O(1 + h_{ijk})$ (в случае отсутствия у задачи входных сообщений сложность итерации цикла константна). Количество итераций равно K_{ij}, где K_{ij} — количество задач j-го раздела i-го ядра. Следовательно, сложность цикла по задачам j-го раздела i-го ядра имеет порядок $O(\sum_{k=1}^{K_{ij}}(1 + h_{ijk}))$.

Одна итерация цикла по разделам, привязанным к ядру, содержит цикл по задачам раздела, порождение 1 автомата и 4-х каналов, 6 операций отождествления для каналов и 3 операции отождествления для переменных. Таким образом, сложность одной итерации цикла по разделам, привязанным к i-му ядру, имеет порядок $O(1 + \sum_{k=1}^{K_{ij}}(1 + h_{ijk}))$. Пусть количество разделов, привязанных к i-му ядру, равно M_i, тогда сложность цикла по разделам, привязанным к i-му ядру, имеет порядок $O(\sum_{j=1}^{M_i}(1 + \sum_{k=1}^{K_{ij}}(1 + h_{ijk}))) = O(M_i + \sum_{j=1}^{M_i}(\sum_{k=1}^{K_{ij}}(1 + h_{ijk})))$.

Одна итерация цикла по ядрам содержит цикл по разделам, привязанным к ядру, порождение 1 автомата, 2 операции отождествления для каналов и $3 \cdot \sum_{j=1}^{M_i} N_{ij}^w$ операций установки значений параметров в значения из описания конфигурации, где N_{ij}^w — количество окон для j-го раздела i-го ядра. В случае отсутствия разделов, привязанных к ядру, сложность итерации цикла константна. Количество итераций в цикле по ядрам равно N. Таким образом, сложность цикла по ядрам имеет порядок $O(\sum_{i=1}^{N}(1+\sum_{j=1}^{M_i} N_{ij}^w+M_i+\sum_{j=1}^{M_i}\sum_{k=1}^{K_{ij}}(1+h_{ijk})))$. Т.к., с учетом введенных в разделе 1.4.1 обозначений, $\sum_{i=1}^{N}\sum_{j=1}^{M_i} N_{ij}^w = \sum_{i=1}^{M} N_i^w$, $\sum_{i=1}^{N}\sum_{j=1}^{M_i}\sum_{k=1}^{K_{ij}}(1+h_{ijk}) = \sum_{i=1}^{M} K_i+H$, $\sum_{i=1}^{N} M_i = M$, то сложность цикла по ядрам имеет порядок $O(N+\sum_{i=1}^{M} N_i^w+M+\sum_{i=1}^{M} K_i+H)$. Обозначим суммарное количество задач в МВС как K: $K = \sum_{i=1}^{M} K_i$. С учетом этого сложность цикла по ядрам имеет порядок $O(N + M + \sum_{i=1}^{M} N_i^w + K + H)$.

На одной итерации цикла по виртуальным каналам порождается 1 автомат, происходит установка одного параметра в значение из описания конфигурации, выполняется 2 операции отождествления для каналов и одна операция отождествления для для переменных. Следовательно, сложность одной итерации цикла константна. Количество итераций в цикле по виртуальным каналам равно H. Следовательно, сложность этого цикла имеет порядок $O(H)$.

Итого, в обозначениях раздела 1.4.1, сложность построения экземпляра модели МВС по описанию конфигурации имеет порядок $O(N + M + \sum_{i=1}^{M} N_i^w + K + H)$. Поскольку каждый раздел содержит хотя бы одну задачу, то $M \leqslant K$ и формула порядка рассматриваемой сложности принимает вид $O(N + \sum_{i=1}^{M} N_i^w + K + H)$.

2.8.2. Сложность прогона экземпляра МВС

Оценим сложность прогона экземпляра модели МВС в зависимости от характеристик соответствующей конфигурации МВС.

Так как все временные характеристики конфигураций МВС представлены целыми числами, то дискретные переходы в автоматах могут происходить только в целочисленные моменты модельного времени. В худшем случае число моментов модельного времени, содержащих дискретные переходы, равно длительности интервала планирования в квантах времени.

По построению автомата-модели планировщика ядра в один момент модельного времени может произойти не более, чем 5 дискретных переходов. Это максимальное количество переходов, которые могут быть выполнены между непрерывными переходами, соответствующими продвижению модельного времени на длительность окна текущего раздела. Следовательно, суммарно по всем моделям планировщиков ядер может произойти не более, чем $5 \cdot N$ переходов. В переходах автомата-модели планировщика ядра не используются функции, сложность которых зависит от характеристик конфигурации МВС. Поэтому общая сложность выполнения всех происходящих одновременно переходов в моделях планировщиков ядер имеет порядок $O(N)$. По построению автомата-модели планировщика ядра переходы не могут происходить в модели ядра, если к нему не привязан ни один раздел. Поэтому число моделей планировщика ядра, в которых происходят переходы, не превышает числа разделов, и сложность выполнения всех происходящих одновременно переходов в моделях планировщиков ядер имеет порядок $O(M)$.

По построению автомата-модели виртуального канала в один момент модельного времени может произойти не более, чем 2 дискретных перехода. Следовательно, суммарно по всем моделям виртуальных каналов может произойти не более, чем $2 \cdot H$ переходов. В переходах автомата-модели виртуального канала не используются функции, сложность которых зависит характеристик конфигурации МВС. Поэтому общая сложность выполнения всех происходящих одновременно переходов в моделях виртуальных каналов имеет порядок $O(H)$.

По построению автоматов-моделей планировщиков работ раздела, в модели планировщика работ i-го раздела в один момент модельного момент времени может произойти не более, чем $K_i + c$ переходов, где c — константа, определяющаяся устройством конкретной модели планировщика. При этом K_i переходов происходит при одновременной готовности работ всех задач раздела. В некоторых переходах автоматов-моделей планировщиков работ используются функции get_job, реализующие один проход по массиву задач раздела, и, следовательно, имеющие сложность порядка $O(K_i)$. Сложность остальных использующихся функций не зависит от характеристик конфигурации МВС. Поэтому сложность выполнения всех переходов, происходящих в один момент модельного времени в модели планировщика работ i-го раздела, имеет порядок $O(K_i^2)$ (с учетом того, что $K_i > 0 \ \ \forall i$). Суммарная по всем моделям планировщиков работ разделов сложность выполнения всех переходов, происходящих в один момент модельного времени, имеет порядок $O(\sum_{i=1}^{M} K_i^2)$. Поскольку $K_1^2 + ... + K_M^2 \leqslant (K_1 + ... + K_M)^2$, то эту оценку можно записать как $O((\sum_{i=1}^{M} K_i)^2) \ = \ O(K^2)$.

По построению автомата-модели функциональной задачи в модели i-й задачи в один момент модельного времени может произойти не более, чем $H_i + 6$ переходов, где H_i — число входных сообщений i-й задачи. При этом H_i переходов происходит при моделировании одновременного получения всех входных сообщений задачи. В некоторых переходах автоматов-моделей задач используются функции $allDataReady()$, $resetData()$, реализующие один проход по массиву входных сообщений задачи, и, следовательно, имеющие сложность $O(H_i)$. Сложность остальных использующихся функций не зависит от характеристик конфигурации МВС. Поэтому сложность выполнения всех переходов, происходящих в один момент модельного времени в модели функциональной задачи, составляет $O(H_i^2 + 1)$. Если задача не имеет входных сообщений ($H_i = 0$), то выполнение всех таких переходов имеет константную сложность. Суммарная по всем моделям задач сложность выполнения всех переходов, происходящих в один момент модельного времени, имеет порядок $O(K + \sum_{i=1}^{K} H_i^2)$. Поскольку $\sum_{i=1}^{K} H_i = H$ и $H_1^2 + ... + H_K^2 \leqslant (H_1 + ..H_K)^2$, то рассматриваемая сложность имеет порядок $O(K + H^2)$.

Итого, суммарная сложность выполнения всех переходов, происходящих в один момент времени в модели МВС, имеет порядок $O(M + H + K^2 + K + H^2) = O(K^2 + H^2)$, т.к. $M \leqslant K$ (количество разделов не превышает количество задач). Следовательно, сложность прогона экземпляра модели имеет порядок $O(L \cdot (M + K^2 + H^2)) = O(L \cdot (K^2 + H^2))$.

Глава 3. Доказательство корректности моделей МВС

При работе над данной главой диссертации использованы следующие публикации автора, в которых, согласно Положению о присуждении ученых степеней в МГУ, отражены основные результаты, положения и выводы исследования:

Глонина А. Б., Балашов В. В. О корректности моделирования модульных вычислительных систем реального времени с помощью сетей временных автоматов // Моделирование и анализ информационных систем. — 2018. — Т. 25, № 2. — С. 174–192. (Glonina A. B., Balashov V. V. On the correctness of real-time modular computer systems modeling with stopwatch automata networks // Automatic Control and Computer Sciences. — 2018. — Vol. 52, № 7. — P. 817–827.)

3.1. Понятие корректности моделей МВС и метод ее обоснования

В разделе 1 описана схема проверки выполнения ограничений реального времени для заданной конфигурации МВС на основе анализа соответствующей ВД МВС. Для применения этой схемы необходимо, чтобы ВД, получаемые при прогоне моделей, соответствовали ВД функционирования целевых систем. Однако на этапе проектирования МВС отсутствует возможность экспериментального сравнения ВД моделей и ВД целевых систем. В то же время стандарты на МВС содержат требования к функционированию таких систем. Среди этих требований присутствуют требования, представляющие собой ограничения на порядок происходящих в МВС событий и длительность интервалов между ними, то есть применимые к ВД МВС. В данной работе корректность моделей МВС рассматривается не с точки зрения соответствия ВД этих моделей недоступным ВД МВС, а с точки зрения соответствия ВД моделей МВС требованиям к МВС, представимым в виде ограничений на ВД. Именно такие требования существенны при проверке выполнения ограничений реального времени.

Согласно предложенному в разделе 1 подходу, ВД МВС с заданной конфигурацией может быть однозначно получена из ВД ее модели посредством выделения из ВД модели событий, существенных для проверки ограничений реального времени. Будем проверять выполнение требований к ВД МВС на уровне ВД моделей и соответствующим образом (через выполнение требований) введем в данной главе понятие корректности ВД модели МВС. Также введем понятие детерминированности модели МВС. Введем понятия корректности для моделей функционирования МВС всех уровней абстракции и докажем, что все предложенные в работе модели корректны.

Из стандартов на МВС с архитектурой ИМА [45, 119] автором были выделены требования, применимые к ВД моделей, то есть представимые в виде ограничений на порядок происходящих в моделях событий и на длительность интервалов между ними. Кроме того, были выделены применимые к ВД моделей требования детерминированности, соответствующие стандартам на МВС и, как правило, учитываемые на этапе проектирования (например, в работах [19, 20, 48, 120]). Все выделенные требования приведены ниже, в разделах 3.2.2 и 3.3.

В рамках данной работы *ВД модели МВС* будем считать *корректной*, если для этой ВД выполнены все выделенные требования.

Согласно предположениям, введенным в разделе 1.4.2, для МВС с заданной конфигурацией ВД ее функционирования определяется однозначно. Поэтому и ВД модели МВС с заданной конфигурацией должна определяться однозначно. Модель МВС с заданной конфигурацией (то есть экземпляр сети автоматов) является *детерминированной*, если ВД, построенные по всем возможным вычислениям этой сети, эквивалентны (определение эквивалентности ВД сети автоматов было дано в разделе 2.3). Таким образом, детерминированность модели позволяет использовать любое вычисление сети автоматов для проверки выполнения ограничений реального времени, в отличие от подхода формальной верификации моделей, предполагающего анализ всех вычислений. *Параметризованная модель МВС* является *детерминированной*, если любой ее экземпляр, построенный для произвольной конкретной конфигурации МВС, является детерминированным.

Модель МВС с заданной конфигурацией будем считать *корректной*, если она является детерминированной и все возможные ее ВД корректны. *Параметризованную модель МВС* будем считать *корректной*, если она является детерминированной, и любой ее экземпляр, построенный для произвольной конкретной конфигурации МВС, является корректным.

Далее под требованием к модели будем понимать требование к ее ВД. Под требованием к модели компонента МВС — требование к последовательностям событий синхронизации, происходящих с участием этой модели. При этом в общем случае требование к некоторой модели компонента МВС должно выполняться при любом поведении других моделей, взаимодействующих с этой моделью.

Таким образом, модель МВС с заданной конфигурацией корректна, если она детерминирована и удовлетворяет всем выделенным требованиям, то есть все ее ВД эквиваленты и для них все выделенные требования выполняются.

Модель компонента МВС считается корректной, если она удовлетворяет всем выделенным для нее требованиям.

Обобщенная модель функционирования МВС считается корректной, если корректны все параметризованные модели МВС, формируемые на ее основе из корректных моделей компонентов МВС.

В перспективе набор требований к моделям компонентов МВС и моделям МВС в целом может быть расширен. Вместе с этим будут уточнены определения корректности моделей.

Модель МВС с заданной конфигурацией строится при помощи приведенного в разделе 2.5 алгоритма по описанию этой конфигурации, на основе параметризованной сети автоматов, реализующей обобщенную модель МВС. Обоснование корректности всех моделей (т.е. экземпляров сетей автоматов), формируемых согласно предложенному методу, выполняется следующим образом.

1. Для каждого базового типа автомата, входящего в состав обобщенной модели МВС, на основе стандартов на МВС [43, 45, 119] выделяется набор требований, которым должны удовлетворять все параметризованные автоматы, реализующие данный базовый тип. Выделенные автором требования приведены в разделе 3.2.2.

2. Доказывается, что если для всех параметризованных автоматов, входящих в состав параметризованной модели МВС, выполняются требования, сформулированные для соответствующих базовых типов автоматов, то параметризованная модель МВС удовлетворяет всем выделенным требованиям корректности к модели в целом, а также является детерминированной. Такие доказательства приведены в разделах 3.3, 3.4.

3. Проверяется выполнение требований к параметризованным автоматам, входящих в состав параметризованной модели МВС. Проверка выполнения этих требований проводится автоматически с помощью верификатора.

4. На основании пунктов 2 и 3 делается вывод о корректности параметризованной модели МВС.

Чтобы при добавлении в параметризованную модель нового параметризованного автомата полученная параметризованная модель была корректной, достаточно убедиться (с использованием верификатора), что все требования, сформулированные для соответствующего базового типа автомата, для данного автомата выполняются, то есть выполнить действия п. 3 для добавляемого автомата. Действия п.п. 1 и 2 производятся однократно и были выполнены автором для предложенной обобщенной модели функционирования МВС. Действия п. 3 были выполнены для всех построенных автором параметризованных автоматов.

Также было проверено выполнение ряда требований корректности, специфичных для отдельных построенных параметризованных автоматов.

С учетом описанного выше подхода к обоснованию корректности формируемых моделей МВС, а также утверждений, доказанных в разделах 3.2—3.4, справедливо следующее утверждение:

Предложенная в настоящей работе обобщенная модель функционирования МВС является корректной.

3.2. Верификация моделей компонентов МВС

3.2.1. Подход к верификации параметризованных автоматов на основе автоматов-наблюдателей

Рассматриваемые в данной работе требования к моделям МВС представляют собой ограничения на порядок происходящих в моделях событий и на длительность интервалов между ними. Согласно разделу 2.2 под событием понимается событие синхронизации автоматов по некоторому каналу.

Модель компонента МВС является параметризованным автоматом с остановкой таймеров. Одним из способов верификации автоматов без параметров является представление требований

в виде формул темпоральных логик и последующая проверка выполнения этих формул. Классические логики LTL [121] и CTL [88], а также логика CTL* [122], являющаяся их обобщением, не позволяют описывать требования, содержащие количественные ограничения на время (например, ограничение на длительность нахождения модели в определенной локации: модель функциональной задачи должна находиться в локации, соответствующей выполнению очередной работы, не более N единиц времени, где N — максимальное время выполнения работы). Для формализации подобных ограничений перечисленные логики имеют временные расширения, например, описанные в [123] и [84]. В то же время, ни логики LTL, CTL и CTL*, ни их временные расширения, не позволяют описать требования вида: «Все последовательности событий в модели должны удовлетворять регулярному выражению $(ab)*\ \ c$ » [124], где a, b, c — типы событий, и выражение $(ab)*$ означает, что последовательность (ab) может повторяться произвольное число раз. В то же время, большинство требований из раздела 3.2.2 имеет подобный вид. Например корректная последовательность синхронизаций, соответствующих выполнению одной работы, должна иметь вид: $exec\ (preempt,\ exec)*\ \ finished$. Существуют расширения классических темпоральных логик, позволяющие формализовать и проверять подобные свойства (например, [124, 125] — расширения CTL и [126] — расширение LTL), но эти расширения не позволяют формализовать ограничения на длительности интервалов между событиями.

Другим подходом к верификации автоматов является подход на основе автоматов-наблюдателей (observer automata) [92]. Для проверки некоторого требования к параметризованному автомату строится сеть автоматов, состоящая из данного автомата и автомата-наблюдателя. Автомат-наблюдатель представляет собой автомат, построенный на основании требования и имеющий интерфейс для взаимодействия (т.е. синхронизаций по каналам и чтения/изменения значений переменных) с исходным автоматом.

Во всех локациях автомата-наблюдателя активны переходы с действиями синхронизации, парными ко всем возможным действиям синхронизации исходного автомата. Таким образом, если некоторая последовательность синхронизаций с участием исходного автомата возможна в некоторой сети автоматов, включающей исходный автомат, то эта последовательность синхронизаций возможна и в сети автоматов, состоящей из исходного автомата и автомата-наблюдателя. Автомат-наблюдатель имеет некоторую «плохую» локацию, такую что любая не удовлетворяющая требованию последовательность синхронизаций гарантированно приводит автомат-наблюдатель в «плохую» локацию.

Параметризованный автомат должен удовлетворять требованиям корректности при всех возможных значениях своих параметров. Поэтому при верификации рассматриваются все возможные комбинации значений параметров из множества допустимых значений. В общем случае, это может привести к проблеме экспоненциального взрыва, но в данной работе предложен подход к сокращению диапазонов перебираемых значений параметров, что делает верификацию практически осуществимой: выделены классы автоматов, для которых диапазоны перебираемых значений параметров могут быть сокращены и доказаны утверждения о допустимости сокращения диапазонов перебираемых значений параметров (всего семь основных утверждений, а также ряд вспомогательных утверждений и лемм). Все разработанные в рамках данной работы автоматы принадлежат этим выделенным классам, поэтому диапазоны перебираемых значений параметров для них были

определены согласно доказанным утверждениям. Формулировки и доказательства утверждений приведены в приложении Б. Если хотя бы для одного набора значений параметров «плохая» локация достижима, то исходный автомат считается некорректным.

Поскольку используемый в данной работе верификатор UPPAAL, работает только с непараметризованными автоматами, то технически перебор всех возможных комбинаций значений из параметров из некоторых диапазонов осуществляется следующим образом. Для верификатора параметр — это общая для автомата-модели и автомата-наблюдателя переменная с заданным диапазоном значений. В автомате-наблюдателе имеется набор служебных приоритетных локаций (в т.ч. такой служебной локацией является начальная локация), при выполнении переходов между которыми происходит установка значений параметров: каждой переменной-параметру присваивается значение, случайным образом выбранное из диапазона значений этого параметра (выбор реализован с использованием конструкции $select$, семантика которой описана на стр. 179). Поскольку указанные служебные локации являются приоритетными, а в автоматах-моделях использование приоритетных локаций в качестве начальных локаций запрещено, то установка значений переменных-параметров выполняется строго до первого перехода в автомате-модели. Сеть автоматов из автомата-модели и автомата-наблюдателя, работающих с переменными-параметрами описанным образом, является непараметризованной сетью автоматов, верифицируемой с помощью UPPAAL. По построению этой сети автоматов каждому корректному набору значений параметров параметризованного автомата-модели соответствует хотя бы одно ее вычисление. И если некоторая локация параметризованного автомата-наблюдателя достижима для хотя бы одного набора корректных значений параметров при совместном функционировании с параметризованным автоматом-моделью, то соответствующая локация будет достижима для описанной непараметризованной сети автоматов. Таким образом, верификация параметризованных автоматов сводится к верификации непараметризованных сетей автоматов.

Кроме того, на порядок событий в исходном автомате могут влиять значения переменных, изменяемых другими автоматами (множество переменных для взаимодействия с другими автоматами определяется интерфейсом соответствующего базового типа автомата). Поэтому в автомате-наблюдателе для каждой локации, соответствующей функционированию исходного автомата (то есть не соответствующей начальной установке параметров и не являющейся «плохой»), имеется переход из нее в нее же, на котором происходит недетерминированное изменение значений таких переменных. Подход к сокращению числа перебираемых при этом значений переменных, позволяющий избежать экспоненциального взрыва, предложен в разделе Б.5 приложения Б. Если хотя бы для одной последовательности значений переменных «плохая» локация достижима, то исходный автомат считается некорректным.

Проверка выполнения требования сводится к проверке недостижимости «плохой» локации. В общем случае задача проверки достижимости в сетях автоматов с остановкой таймеров алгоритмически неразрешима [94], однако в [97—99] доказано, что для частного случая, соответствующего рассматриваемым в данной работе моделям (детерминированность планировщиков, использование заранее известных оценок времени выполнения работ и времени передачи сообщений), эта проблема разрешима. При использовании верификатора UPPAAL гарантируется [63], что процесс верификации завершается всегда (возможно, с неопределенным результатом). Таким

образом, завершение верификации с результатом «плохая локация недостижима» говорит о выполнении требования, завершение верификации с результатом «плохая локация достижима» — о невыполнении требования (т.е. модель не может быть использована), завершение верификации с неопределенным результатом — о том, что модель не может быть использована, так как нет подтверждения выполнения для нее требования корректности. В случае завершения верификации с неопределенным результатом, параметризованный автомат необходимо перестроить так, чтобы результат верификации был определен. В работах [97—99] показано, что для рассматриваемого класса МВС автоматы, моделирующие их компоненты, можно построить таким образом, что задача их верификации алгоритмически разрешима. В перечисленных работах модели МВС построены так, что значения всех таймеров ограничены, и проблема верификации алгоритмически разрешима [96].

3.2.2. Проверка выполнения требований корректности к моделям компонентов МВС

В данном разделе приведены требования к параметризованным автоматам, моделирующим компоненты МВС. Для одного требования описана логика построения соответствующего автомата-наблюдателя и приведен этот автомат-наблюдатель. Для остальных требований приведены только формулировки. Автоматы-наблюдатели и логика их построения для этих требований приведены в приложении В. Для корректности моделей МВС, формируемых согласно предложенному методу, необходимо выполнение всей совокупности требований: если хотя бы одно требование не выполнено, то модели некорректны.

При построении автоматов-наблюдателей, соответствующих некоторым из приведенных ниже требований, использовались предположения о том, что выполнены другие требования. Например, при построении всех автоматов-наблюдателей для проверки выполнения требований к моделям планировщиков задач разделов использовалось предположение о том, что синхронизация по каналу $ready$ возможна, только если хотя бы одна из переменных is_ready равна 1. Это предположение соответствует выполнению требования к моделям функциональных задач. Также выполнение некоторых требований является следствием выполнения других требований. Например, выполнение требования 6 к моделям планировщиков задач разделов является следствием выполнения требований 1, 4 и 5 к этим моделям.

На рисунке 3.1 приведен граф логических зависимостей между требованиями к моделям базовых типов компонентов МВС. Эти зависимости отражают ход рассуждений при построении автоматов-наблюдателей. Порядок же проверки выполнения требований может быть произвольным. Использованные на рисунке 3.1 обозначения требований состоят из имени базового типа автоматов (например, TS) и номера требования.

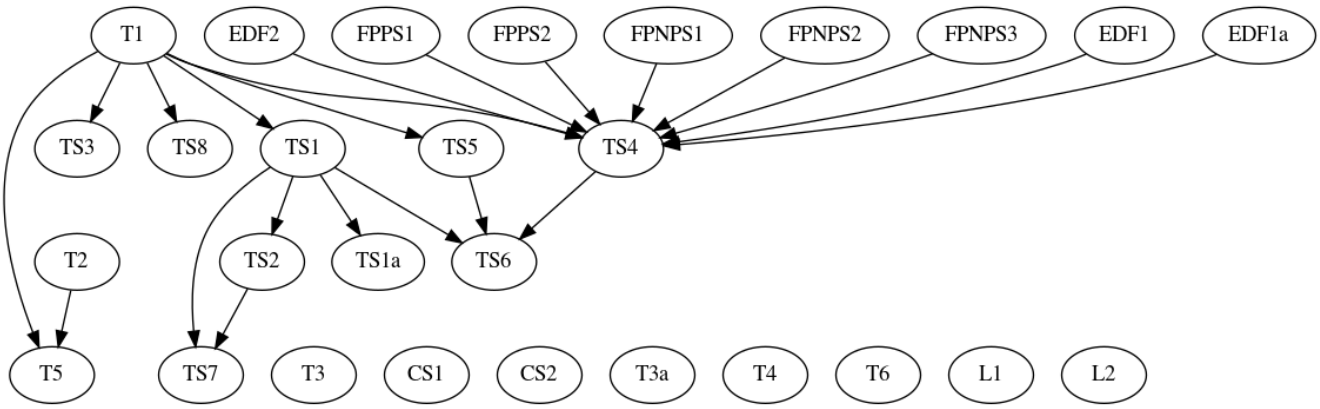

Рисунок 3.1 — Логические зависимости между требованиями к моделям базовых типов компонентов МВС

Требования к моделям планировщиков раздела

Ниже приведены требования к моделям планировщиков раздела, то есть параметризованным автоматам, реализующим базовый тип автоматов **`TS`**.

По построению модели МВС (см. раздел 2.5) модель планировщика раздела взаимодействует с моделью планировщика того ядра, к которому привязан раздел, а также с моделями функциональных задач, принадлежащих этому разделу. Поэтому в приведенных ниже требованиях к модели планировщика раздела речь идет только о задачах, принадлежащих этому разделу.

Требование 1. *Для любого раздела верно, что в каждый момент времени на вычислителе может выполняться не более одной работы этого раздела* [45].

В терминах моделируемых событий данное требование может быть сформулировано следующим образом: *Для любого раздела верно, что если некоторая работа этого раздела была поставлена на выполнение, то любая другая работа этого раздела может быть поставлена на выполнение только после вытеснения, либо завершения первой работы.*

Переформулируем требование в терминах синхронизаций с участием автомата, моделирующего планировщик раздела, и построим автомат-наблюдатель для проверки этого требования. Постановке на выполнение работы i-й задачи соответствует синхронизация по каналу $exec_i$, вытеснению — $preempt_i$, завершению — $finished$.

Все последовательности синхронизаций с участием модели планировщика раздела, удовлетворяющие требованию, представляют собой повторяющиеся подпоследовательности следующего вида (здесь и далее i — произвольные номера задач данного раздела, j — некоторый фиксированный в рамках подпоследовательности номер задачи данного раздела):

1) некоторое (возможно, нулевое) количество синхронизаций по любым каналам, кроме $exec_i$;
2) синхронизация по каналу $exec_j$ (постановка некоторой работы j-й задачи на выполнение);
3) некоторое (возможно, нулевое) количество синхронизаций по любым каналам, кроме $exec_i$, $preempt_i$ и $finished$ (происходящие синхронизации соответствуют событиям во время выполнения работы);

4) синхронизация либо по каналу $preempt_j$ (вытеснение выполняющейся работы), либо по каналу $finished$ (завершение работы).

В случае конечного времени функционирования МВС (что соответствует реальным сценариям) последняя подпоследовательность синхронизаций может быть неполной. Это не приводит к нарушению корректности. Данное замечание относится ко всем последовательностям синхронизаций, соответствующим корректным моделям.

Для того, чтобы последовательность синхронизаций нарушала требование 1, необходимо и достаточно, чтобы она начиналась с подпоследовательности синхронизаций одного из двух видов (P и Q):

(P) — вытеснение не той работы, которая была поставлена на выполнение:

1) конечная последовательность, удовлетворяющая требованию 1, либо пустая последовательность (корректное функционирование планировщика);
2) некоторое (возможно, нулевое) количество синхронизаций по любым каналам, кроме $exec_i$ (события до постановки некоторой работы на выполнение);
3) синхронизация по каналу $exec_j$ (постановка некоторой работы j-й задачи на выполнение);
4) некоторое (возможно, нулевое) количество синхронизаций по любым каналам, кроме $exec_i$, $preempt_i$ и $finished$ (происходящие синхронизации соответствуют событиям во время выполнения работы);
5) синхронизация по каналу $preempt_i$, где $i \neq j$ (вытеснение не той работы, которая была поставлена на выполнение).

(Q) — постановка некоторой работы на выполнение при наличии на вычислителе другой работы:

1) конечная последовательность, удовлетворяющая требованию 1, либо пустая последовательность (корректное функционирование планировщика);
2) некоторое (возможно, нулевое) количество синхронизаций по любым каналам, кроме $exec_i$ (события до постановки некоторой работы на выполнение);
3) синхронизация по каналу $exec_j$ (постановка некоторой работы j-й задачи на выполнение);
4) некоторое (возможно, нулевое) количество синхронизаций по любым каналам, кроме $exec_i$, $preempt_i$ и $finished$ (происходящие синхронизации соответствуют событиям во время выполнения работы);
5) синхронизация по каналу $exec_i$ (постановка некоторой работы на выполнение).

Последовательности синхронизаций, удовлетворяющие требованию 1, могут не удовлетворять другим требованиям. Например, требованию 1 будет удовлетворять последовательность синхронизаций, в которой синхронизация по каналу $exec_i$ выполняется после очередной синхронизации по каналу $sleep$, но до последующей синхронизации по каналу $wakeup$. Эта последовательность соответствует некорректному функционированию планировщика раздела — постановке работы раздела на выполнение вне окна раздела. Однако, этот факт не может привести к ложному признанию модели корректной, так как согласно разделу 3.1 модель считается корректной, если она удовлетворяет *всей совокупности* требований. Так, приведенная последовательность синхронизаций не удовлетворяет требованию 2, приведенному ниже. Если при

функционировании модели планировщика возможно появление такой последовательности, то это будет обнаружено при проверке требования 2, и модель будет признана некорректной.

Теперь, когда указаны все корректные и некорректные последовательности синхронизаций, можно приступить к описанию автомата-наблюдателя, приведенного на рисунке 3.2

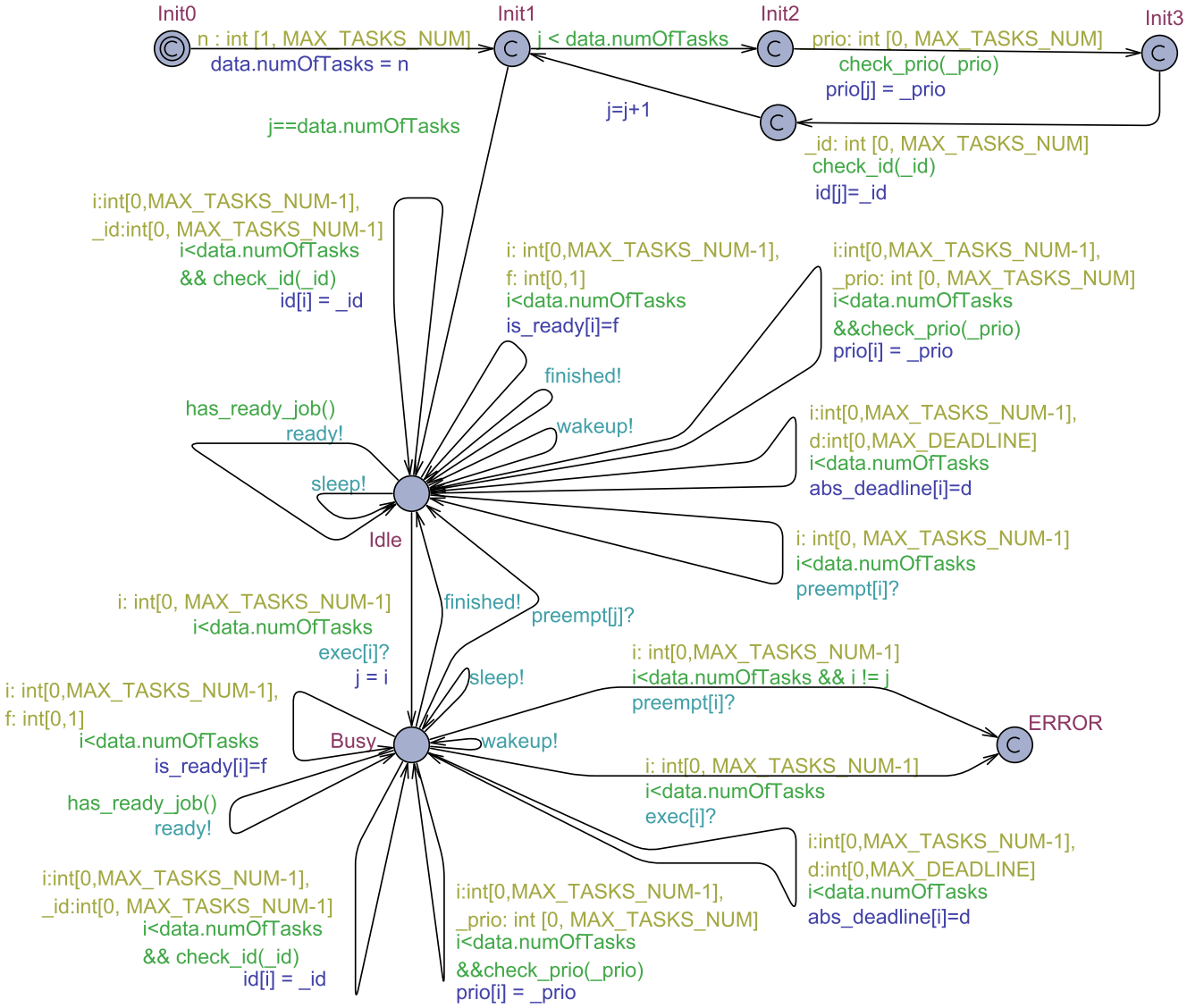

Рисунок 3.2 — Автомат-наблюдатель, соответствующий требованию 1 к модели планировщика раздела

При переходах между локациями $\{Init0, ..., Init3\}$ происходит недетерминированная установка значений параметров и начальных значений переменных автомата: выбирается количество задач, для каждой задачи выбирается уникальный приоритет ($prio$) и уникальный идентификатор (id) (выбор диапазонов значений параметров и переменных описан далее на стр. 71). Уникальность приоритета обеспечивается функцией $check\_prio$, идентификатора — $check\_id$. Данная функция, как и все другие функции, используемые на переходах автоматов, написана на языке, поддерживаемом верификатором UPPAAL; применение этого языка при моделировании не увеличивает выразительность выбранного математического аппарата и не влияет на возможность верификации, но позволяет кратко записывать громоздкие выражения. Если для верифицируемого параметризованного автомата известно, что в его системе переходов параметры «приоритеты

задач» не используются, то установку их значений целесообразно опустить для ускорения последующей верификации. Например, приоритеты задач не используются в разработанной автором модели планировщика, работающего по стратегии EDF.

Описанная схема установки значений параметров является одинаковой для всех автоматов-наблюдателей, используемых для проверки выполнения требований к моделям планировщиков раздела.

Помимо локаций $\{Init0, ..., Init3\}$, в автомате-наблюдателе присутствуют локации $Busy$ и $Idle$, соответствующие выполнению некоторой работы и отсутствию выполняющихся работ, а также «плохая» локация $ERROR$. Из локаций $Busy$ и $Idle$ имеются переходы по всем определенным для базового типа TS действиям синхронизации. Переходы между локациями $Busy$ и $Idle$ соответствуют описанным выше удовлетворяющим требованию 1 последовательностям синхронизаций, переходы в локацию $ERROR$ — описанным выше не удовлетворяющим требованию 1 последовательностям синхронизаций. Также интерфейс базового типа TS содержит наборы переменных is_ready, $abs\_deadline$. Поэтому для каждой из локаций $Busy$ и $Idle$ имеются переходы в ту же локацию с недетерминированным изменением этих переменных. Синхронизации по каналу $ready$ возможны только при наличии хотя бы одной готовой работы, то есть работы, для которой соответствующая переменная is_ready равна 1 (такая проверка реализована в функции $has\_ready\_job()$; эта функция используется и в последующих автоматах-наблюдателях). Это ограничение на синхронизации по каналу $ready$ соответствует выполнению требования 1 к моделям функциональных задач, приведенному ниже. То есть прошедшая верификацию модель планировщика работ раздела будет гарантированно соответствовать требованию 1 к моделям планировщиков работ раздела, если взаимодействующие с ней модели функциональных задач соответствуют требованию 1 к моделям функциональных задач. Для хранения номера задачи, работа которой поставлена на выполнение, используется переменная j.

Покажем, как для верификации выбираются значения параметров и переменных в соответствии с рассуждениями, приведенными в приложении Б, на примере разработанной автором модели планировщика с фиксированными приоритетами и вытеснением (описание модели см. в разделе А.2 приложения А). Модель работает с одним массивом и удовлетворяет условиям 1—6 и 8—11 определения класса автоматов У6. Автомат-наблюдатель также удовлетворяет условиям 1—6 и 8—11 определения класса автоматов У6. Условие 7 определения класса автоматов У6 выполняется по построению автомата-наблюдателя. Поэтому, согласно рассуждениям на стр. 177, достаточные для верификации значения параметра-размера массива определяются в соответствии с утверждением 6. Суммарное количество функций и индексных переменных в модели равно 2. Суммарное количество функций и индексных переменных в автомате-наблюдателе равно 2. Поэтому, согласно утверждению 6, при верификации достаточно рассматривать массивы размера не более, чем 4. В модели не используются временные и индексные параметры. Автомат-наблюдатель и автомат-модель удовлетворяют условиям 1 и 3 утверждения 9, поэтому, согласно рассуждениям на стр. 201, значения элементов массивов $prio$, $abs\_deadline$ и id (т.е. переменных $prio_i$, $abs\_deadline_i$, id_i) выбираются при верификации в соответствии с утверждением 9. Максимальный рассматриваемый при верификации размер каждого из этих массивов равен 4, поэтому,

согласно утверждению 9, в качестве возможных значений элементов указанных массивов перебираются значения от 1 до 4.

Выбор значений параметров в остальных автоматах-наблюдателях, используемых для проверки выполнения требований к моделям планировщиков раздела, осуществляется аналогично.

Для остальных требований к моделям планировщиков раздела приведены только формулировки. Соответствующие автоматы-наблюдатели приведены в приложении В.1.

Построенный автомат-наблюдатель позволяет выявлять подпоследовательности синхронизаций вида P (вытеснение не той работы, которая была поставлена на выполнение), поэтому он используется в т.ч. в процессе проверки выполнения требования 1а.

Требование 1а. *Для любого раздела верно, что в некоторый момент времени вытеснена может быть только выполняющаяся в этот момент работа.*

Требование 2. *Для любого раздела верно, что работы данного раздела могут выполняться только в границах окон этого раздела* [119].

Некоторые требования для упрощения построения автоматов-наблюдателей могут быть разбиты на несколько вспомогательных требований. Для выполнения исходного требования достаточно выполнения всех вспомогательных требований.

Требование 2 может быть представлено в виде совокупности следующих трех вспомогательных требований:

2.1. *Планировщик раздела может поставить работу на выполнение только в рамках окна этого раздела.*

2.2. *Для любого раздела верно, что если в некоторый момент времени на вычислителе выполняется работа этого раздела и происходит закрытие текущего окна этого раздела, то некоторая работа должна быть мгновенно вытеснена, либо завершена*. Данное вспомогательное требование не специфицирует, какая именно работа должна быть вытеснена, либо завершена. Напомним, что в требованиях к моделям планировщиков раздела речь идет только о работах, принадлежащих разделу, с которым работает планировщик.

2.3. *Для любого раздела верно, что в каждый момент времени на вычислителе может выполняться не более одной работы этого раздела.* Таким образом, при закрытии окна раздела необходимо вытеснить не более одной работы этого раздела. Заметим, что данное вспомогательное требование совпадает с требованием 1. Автомат-наблюдатель для проверки его выполнения уже был описан ранее. Этот автомат-наблюдатель позволяет проверить в том числе и то, что только выполняющаяся работа может быть вытеснена.

Обоснуем достаточность выполнения требований 2.1—2.3 для выполнения требования 2. Для этого обоснуем эквивалентное утверждение о том, что из нарушения требования 2 следует нарушение хотя бы одного из требований 2.1—2.3. Действительно, если нарушено требование 2, то либо нарушено требование 2.1 (некоторая работа была поставлена на выполнение вне окна соответствующего раздела), либо нарушено требование 2.2 (в момент закрытия окна раздела при наличии выполняющейся работы никакая работа не была вытеснена или завершена, в связи с чем выполняющаяся работа продолжила выполнение за пределами окна раздела), либо нарушено требование 2.3 (в момент закрытия окна раздела на вычислителе выполнялось несколько работ этого

раздела, так что при завершении или вытеснении одной из них остальные продолжили выполняться за пределами окна раздела).

Требование 3. *Для любого раздела верно, что планировщик данного раздела может поставить на выполнение только готовую работу этого раздела [45].*

Требование 4. *Для любого раздела верно, что в момент времени, когда осуществляется выбор работы для постановки на выполнение, этот выбор делается однозначно, и постановка на выполнение осуществляется мгновенно.*

Данное требование в явном виде не содержится в стандартах на МВС, однако оно соответствует одному из ключевых предположений данной работы (правомерность использования этого предположения обоснована в разделе 1.4.2).

Требование 4 может быть конкретизировано и уточнено совокупностью следующих вспомогательных требований:

4.1. *Работа может быть поставлена на выполнение только вследствие следующих событий: открытие окна раздела, появление новой готовой работы, завершение некоторой работы. При этом постановка работы на выполнение происходит мгновенно, т.е. одновременно с событием-причиной.*

4.2. *Если открывается окно раздела и имеются готовые работы этого раздела, то на выполнение мгновенно должна быть поставлена некоторая работа.*

4.3. *Если завершилась некоторая работа раздела, имеются готовые работы этого раздела, окно раздела открыто и в данный момент не происходит закрытие окна раздела, то на выполнение мгновенно должна быть поставлена некоторая работа.*

4.4. *Если в некоторый момент времени появляется новая готовая работа, то выбор, должна ли какая-либо работа быть поставлена на выполнение в этот момент, осуществляется однозначно.* При этом, если работа должна быть поставлена на выполнение, то постановка на выполнение выполняется мгновенно согласно требованию 4.1.

4.5. *Если в некоторый момент времени некоторая работа должна быть поставлена на выполнение, то выбор работы для постановки на выполнение осуществляется однозначно.*

С учетом приведенной детализации, будем считать, что требование 4 выполнено, если выполнены все требования 4.1—4.5, и требование 4 нарушено, если нарушено хотя бы одно из требований 4.1—4.5.

Требования мгновенности реакции на перечисленные события можно заменить на требования реакции на эти события с фиксированной задержкой. Например: «С момента завершения работы до момента постановки новой работы на выполнение проходит N тактов, где значение N задано в конфигурации». Однако в большинстве работ по планированию вычислений в МВС, например в [19, 20, 48], данной задержкой пренебрегают, так как работы одного раздела выполняются в общем адресном пространстве и величина задержки мала. Кроме того, модель предполагает, что между окнами разделов могут быть интервалы, не принадлежащие никаким разделам, поэтому такие интервалы могут использоваться для моделирования задержек, необходимых для инициализации окон.

Требования 4.4 и 4.5 заключаются в однозначности реакции модели на некоторое событие. При этом на уровне базового типа автоматов не известно, какая именно ожидается реакция на

то или иное событие в каждом конкретном контексте, например при конкретном наборе готовых работ. Согласно подходу, описанному в разделе 3.2.1, в автомате-наблюдателе происходит переход в «плохую» локацию, если реакция верифицируемого автомата на некоторое событие (синхронизацию по каналу, изменение значения переменной или таймера) не соответствует требованию. Однозначность реакции, т.е. ее одинаковость для двух любых совпадающих до этого события вычислений сети автоматов, такой подход проверить не позволяет. Однако алгоритмы планирования, реализованные в конкретных типах планировщиков раздела, однозначно определяют правила выбора работы для постановки на выполнение, а также действия планировщика при появлении новой готовой работы. Эти правила и действия зависят только от текущего состояния модели планировщика и не зависят от предыстории. Согласно ограничениям, введенным в разделе 1.4.2, иные алгоритмы планирования в данной работе не рассматриваются. Для каждого из конкретных типов планировщиков раздела, рассматриваемых в данной работе, ниже (в разделе 3.2.2) сформулированы требования, однозначно определяющие правила выбора работы для постановки на выполнение, и действия планировщика при появлении новой готовой работы. Также ниже (в приложении В.5) приведены автоматы-наблюдатели для проверки этих требований. Требования, специфичные для конкретных типов планировщиков, уточняют требования 4.4 и 4.5, и из выполнения требований к моделям конкретных типов планировщиков следует выполнение требований 4.4 и 4.5.

Требование 5. *Для любого раздела верно, что любая выполняющаяся работа этого раздела может быть вытеснена только в двух случаях: либо при закрытии окна данного раздела, либо при появлении новой готовой работы. При этом вытеснение происходит мгновенно* [45].

Из выполнения требований 1, 4.1, 4.4 и 5 следует выполнение требования 6:

Требование 6. *Для любого раздела верно, что если в некоторый момент времени на вычислителе выполняется работа раздела и появляется новая готовая работа этого раздела, которая должна быть поставлена на выполнение, то текущая работа мгновенно вытесняется* [45].

Действительно, из выполнения требований 4.1 и 4.4. следует, что если в некоторый момент времени появляется новая готовая работа, которая должна быть поставлена на выполнение, то она ставится на выполнение мгновенно. Из выполнения требования 1 следует, что перед постановкой на выполнение новой готовой работы текущая выполняющаяся работа должна быть вытеснена. А из выполнения требования 5 следует, это вытеснение происходит мгновенно. Требование 6 будет использовано далее, в разделе 3.4.

Из выполнения требований 1а, 2.2 и 2.3 следует выполнение требования 7:

Требование 7. *Для любого раздела верно, что если в некоторый момент времени на вычислителе выполняется работа раздела и происходит закрытие текущего окна этого раздела, то данная работа мгновенно вытесняется, либо некоторая работа завершается* [45].

В приведенной формулировке указано завершение *некоторой* работы, а не конкретной выполняющейся работы, так как, согласно предложенной в разделе 2.4 модели, для уведомления модели планировщика раздела о завершении любой работы этого раздела используется синхронизация по одному и тому же каналу $finished$. Проверка того, что только выполняющаяся работа может быть завершена, происходит при проверке требования 3 к моделям функциональных задач.

Действительно, из выполнения требования 2.2 следует, что если в некоторый момент времени на вычислителе выполняется работа раздела и происходит закрытие текущего окна этого раздела, то *некоторая* работа мгновенно вытесняется, либо завершается. Из выполнения требования 1а следует, что вытеснена может быть только выполняющаяся работа. А из выполнения требования 2.3 (единственности выполняющейся работы) следует однозначность определения вытесняемой работы. Требование 7 будет также использовано далее, в разделе 3.4.

Требование 8. *Для любого раздела верно, что планировщик этого раздела мгновенно реагирует на события открытия и закрытия очередного окна этого раздела* [119]. То есть как только в автомате, функционирующем совместно с автоматом-моделью планировщика раздела, становится активным переход с синхронизацией по каналу $wakeup$, либо по каналу $sleep$, эта синхронизация мгновенно выполняется.

Требование мгновенности можно заменить на требование реакции с фиксированной задержкой (см. пояснение к требованиям 4.1—4.3).

Требования к моделям функциональных задач

Ниже приведены требования к моделям функциональных задач, то есть параметризованным автоматам, реализующим базовый тип автоматов **T**. Автоматы-наблюдатели для проверки этих требований приведены в приложении В.2.

Требование 1. *Для любой задачи верно, что некоторая ее работа может быть поставлена на выполнение, только если эта работа является готовой* [45].

Требование 2. *Для любой задачи верно, что очередная ее работа становится готовой мгновенно после того как становится истинной конъюнкция трех условий: (1) модельное время не меньше левой границы директивного интервала; (2) получены синхронные сообщения от всех соответствующих работ-отправителей (если задача зависит по данным от других задач), (3) модельное время меньше правой границы директивного интервала.*

Стандарт ARINC 653 [45] описывает ряд механизмов взаимодействия работ (буферы, семафоры и др.), с помощью которых могут быть реализованы синхронные зависимости по данным. Эти механизмы позволяют реализовать и более сложные условия готовности работы. Однако, в ряде современных МВС для предсказуемости временных характеристик специально запрещено использование блокирующих средств синхронизации, кроме механизмов ожидания получения синхронных сообщений. Так, согласно работе [127], в программной системе управления шасси современного самолета, являющейся одним из разделов, используется лишь механизм ожидания получения синхронных сообщений. В работах по планированию вычислений, как правило (например, в [19, 20, 64]), рассматриваются лишь указанные в требовании условия готовности.

То, что работа не может быть готовой после своего завершения (даже если все три условия требования 2 выполнены), соответствует выполнению требования 3а, приведенного ниже.

При обосновании выполнения требования 2 для модели задачи также должно быть проверено следующее вспомогательное требование:

Для любой задачи, имеющей входные сообщения, верно, что очередная ее работа, соответствующая некоторому периоду, может использовать лишь те сообщения, которые получены от работ, соответствующих тому же самому периоду.

Требование 3. *Выполняющаяся работа задачи должна завершиться как только станет выполненным хотя бы одно из следующих условий: длительность ее выполнения на вычислительном ядре достигло значения WCET; текущее время достигло правой границы директивного интервала этой работы. Работа не может завершиться, если ни одно из двух указанных условий не выполнено. Работа также не может завершиться, если она не является выполняющейся*. Под завершением в данном случае понимается оповещение планировщика о прекращении выполнения находящейся на вычислительном ядре работы.

Данное требование соответствует введенному в разделе 1 предположению о том, что выполняющаяся работа либо завершается принудительно из-за опоздания, либо длительность ее выполнения на вычислительном ядре равна WCET.

Под завершением работы понимается уведомление планировщика о том, что выполнение текущей работы окончено и, следовательно, требуется определить работу, которая будет поставлена на выполнение далее.

Требование 3а. *Для любой работы любой задачи верно, что с момента ее завершения до момента окончания соответствующего периода задачи эта работа не может быть готовой.*

Требование 4. *Для любой задачи верно, что если эта задача имеет выходные сообщения, то каждая ее работа должна однократно послать сообщения через все выходные виртуальные каналы строго в момент завершения работы при условии штатного завершения. Если работа не успела штатно завершиться, то отправка сообщений не происходит.* Согласно разделу 1.1, в данной работе рассматриваются лишь синхронные сообщения.

Следствием выполнения требований 1 и 2 является выполнение требования 5:

Требование 5. *Для любой задачи верно, что любая ее работа может быть поставлена на выполнение только в рамках своего директивного интервала.*

Действительно, согласно требованию 1, на выполнение может быть поставлена только готовая на момент постановки работа. А согласно требованию 2, работа может быть готовой только в границах своего директивного интервала.

Требование 6. *Для любой задачи верно, что значение абсолютной правой границы директивного интервала ее очередной работы рассчитывается в соответствии с его определением, то есть значение абсолютной правой границы директивного интервала очередной работы равно времени начала соответствующего этой работе периода, увеличенному на значение относительной правой границы ее директивного интервала*.

Требования к моделям планировщиков ядра

Ниже приведены требования к моделям планировщиков ядра, то есть параметризованным автоматам, реализующим базовый тип автоматов **CS**. Автоматы-наблюдатели для проверки этих требований приведены в приложении В.3.

В разделе 1.4.1 были введены ограничения корректности, которым должно удовлетворять расписание окон. Проверку выполнения этих ограничений целесообразно выполнять не в модели планировщика ядра, а при построении модели, соответствующей заданной конфигурации МВС. При верификации автомата-модели планировщика ядра будем считать, что ее параметры всегда задают корректное расписание окон.

Единственным назначением модели планировщика ядра является уведомление моделей планировщиков разделов об открытии и закрытии соответствующих разделам окон согласно заданному в конфигурации расписанию, поэтому к моделям планировщика ядра сформулировано лишь два требования:

Требование 1. *Для любого ядра верно, что для любого окна раздела планировщик ядра однократно оповещает планировщик раздела, соответствующего данному окну, об открытии этого окна строго в момент открытия, определенный в расписании.*

Требование 2. *Для любого ядра верно, что для любого окна раздела планировщик ядра однократно оповещает планировщик раздела, соответствующего данному окну, о закрытии этого окна строго в момент закрытия, определенный в расписании.*

Требования к моделям виртуальных каналов

Ниже приведены требования к моделям виртуальных каналов, то есть параметризованным автоматам, реализующим базовый тип автоматов **L**. Автоматы-наблюдатели для проверки этих требований приведены в приложении В.4.

Требование 1. *Для любого виртуального канала верно, что задержка на передачу сообщения через виртуальный канал строго равна значению, указанному в конфигурации системы.*

Данное требование соответствует предположению о фиксированной задержке на передачу, сформулированному и обоснованному в разделе 1.4.2.

Требование 2. *Для любого виртуального канала и любого сообщения, передаваемого по нему, верно, что получатель оповещается о получении сообщения ровно один раз, в момент окончания передачи.*

Требование единственности оповещения соответствует логике функционирования виртуального канала (так, например, в стандарте AFDX [43] описан механизм обработки дублей сообщений, предполагающий получение приложением единственного сообщения).

Требования к моделям конкретных типов компонентов МВС

Выше в данном разделе для каждого базового типа автоматов приведены требования, которым должны удовлетворять все параметризованные автоматы, реализующие этот базовый тип автоматов (соответствующий базовому типу компонента МВС). Также существуют требования к параметризованным автоматам, моделирующим конкретные типы компонентов МВС. Эти требования специфичны для каждого конкретного типа компонента МВС. В данном разделе приведены требования, специфичные для моделей различных типов планировщиков раздела, и автоматы-наблюдатели для проверки этих требований. Эти требования и автоматы-наблюдатели для их проверки применимы к любым параметризованным автоматам, моделирующим планировщики соответствующих конкретных типов. Поэтому если для решения некоторой задачи необходимо вместо разработанной автором работы модели планировщика использовать другую модель планировщика того же конкретного типа, то пользовательская модель должна удовлетворять требованиям, соответствующим этому конкретному типу.

Требования к модели планировщика с фиксированными приоритетами и вытеснением

Требование 1. *Для любого раздела верно, что если выполнением работ этого раздела управляет планировщик с фиксированными приоритетами и вытеснением, то для постановки на выполнение всегда выбирается работа с максимальным среди всех готовых работ приоритетом*.

Требование 2. *Для любого раздела верно, что если выполнением работ этого раздела управляет планировщик с фиксированными приоритетами и вытеснением, то при появлении новой готовой работы с максимальным среди всех готовых работ приоритетом, некоторая работа мгновенно ставится на выполнение при условии того, что окно этого раздела открыто*.

Требования к модели планировщика, работающего по стратегии EDF с вытеснением

Требование 1. *Для любого раздела верно, что если выполнением работ этого раздела управляет планировщик, работающий по стратегии EDF с вытеснением, то для постановки на выполнение всегда выбирается готовая работа с наименьшей правой границей директивного интервала*.

Требование 1 не определяет порядок постановки на выполнение готовых работ, имеющих одинаковые правые границы директивных интервалов. Однако, алгоритм планирования должен быть детерминированным, поэтому существуют различные модификации алгоритма EDF. Так, в [128] готовые работы с одинаковыми правыми границами директивных интервалов планируются согласно дисциплине FIFO. Однако, в случае наличия таких работ, становящихся готовыми одновременно, алгоритм является недетерминированным (это возможно при наличии задач, не имеющих входных зависимостей по данным, и имеющих одинаковые директивные интервалы). В [129] среди готовых работ с одинаковыми правыми границами директивных интервалов для постановки на выполнение выбирается та, WCET которой больше, чем у прочих. Однако, в случае наличия работ с одинаковыми WCET алгоритм также является недетерминированным. В [130] каждая задача имеет уникальный идентификатор и среди готовых работ с одинаковыми правыми границами директивных интервалов для постановки на выполнение выбирается та, идентификатор задачи у которой меньше, чем идентификаторы задач у остальных работ. Разработанная

автором модель планировщика, функционирующего по стратегии EDF, имеет такую же как в [130] детерминированную дисциплину планирования, то есть, помимо требования 1, должна удовлетворять следующему требованию:

Требование 1а. *Для любого раздела верно, что если выполнением работ этого раздела управляет планировщик, работающий по детерминированной стратегии EDF с вытеснением, то среди готовых работ с одинаковыми правыми границами директивных интервалов для постановки на выполнение выбирается та, номер задачи у которой меньше, чем у остальных*.

Требование 2. *Для любого раздела верно, что если выполнением работ этого раздела управляет планировщик, работающий по стратегии EDF с вытеснением, то при появлении новой готовой работы с меньшей, чем у всех готовых работ, правой границей директивного интервала, некоторая работа мгновенно ставится на выполнение при условии того, что окно этого раздела открыто*.

Требования к модели планировщика с фиксированными приоритетами без вытеснения

Требование 1. *Для любого раздела верно, что если выполнением работ этого раздела управляет планировщик с фиксированными приоритетами без вытеснения, то для постановки на выполнение всегда выбирается работа с максимальным среди всех готовых работ приоритетом*.

Требование 2. *Для любого раздела верно, что если выполнением работ этого раздела управляет планировщик с фиксированными приоритетами без вытеснения, то любая выполняющаяся работа этого раздела может быть вытеснена только при закрытии окна данного раздела. При этом вытеснение происходит мгновенно.*

Требование 3. *Для любого раздела верно, что если выполнением работ этого раздела управляет планировщик с фиксированными приоритетами без вытеснения, то при появлении новой готовой работы некоторая работа мгновенно ставится на выполнение при условии того, что окно этого раздела открыто и на вычислительном ядре нет выполняющихся работ*.

3.3. Корректность модели МВС в целом

Помимо требований корректности к моделям отдельных компонентов МВС, существуют и требования корректности к модели МВС в целом. Проверить выполнение этих требований автоматизированно с помощью верификатора невозможно, так как в общем случае неизвестно количество автоматов в параметризованной сети автоматов, и задача верификации алгоритмически неразрешима [131]. Кроме того, даже если количество автоматов в сети автоматов известно и соответствует количеству компонентов в реальной МВС, то верифицировать такую сеть автоматов на практике невозможно ввиду экспоненциальной зависимости времени верификации от количества компонентов модели (во Введении приведены численные оценки времени верификации моделей, содержащих несколько десятков компонентов). Поэтому выполнение таких требований

было доказано автором путем строгих логических рассуждений на основе информации о структуре модели (см. разделы 2.4, 2.5), а также на основе выполнения требований к моделям компонентов МВС (см. раздел 3.2) и требований корректности к входным данным модели (см. раздел 1.4.1).

Ниже приведены требования к модели в целом и доказательства их выполнения. В рамках перспективных исследований набор требований может быть расширен.

Требование 1. *Для любых двух задач верно, что если одна задача зависит по данным от другой, то время начала выполнения каждой работы задачи-получателя данных не меньше времени завершения соответствующей работы задачи-отправителя данных плюс время передачи данных.*

Доказательство выполнения данного требования основано на выполнении требований к реализациям базовых типов автоматов **L** (модель виртуального канала) и **T** (модель функциональной задачи). Из выполнения требования 2 к модели функциональной задачи следует, что время начала выполнения очередной работы задачи-получателя не меньше времени получения сообщения от соответствующей работы задачи-отправителя. Из выполнения требования 1 к модели виртуального канала следует, что время получения сообщения равно времени его отправки плюс значение задержки на передачу, указанное в конфигурации. Из выполнения требования 4 к модели задачи следует, что отправка сообщения происходит строго в момент завершения работы задачи-отправителя. Таким образом, из выполнения перечисленных требований к моделям компонентов следует выполнение указанного требования к модели в целом.

Требование 2. *Для любого вычислительного ядра системы верно, что в каждый момент времени на нем выполняется не более одной работы.*

Доказательство выполнения данного требования основано на выполнении требований к реализациям базовых типов автоматов **CS** (модель планировщика ядра) и **TS** (модель планировщика раздела). Из выполнения требования 1 к модели планировщика раздела следует, что в каждый момент времени на ядре может выполняться не более одной работы каждого раздела. Из выполнения требования 2 к модели планировщика раздела следует, что работы некоторого раздела могут выполняться только в границах окон этого раздела. А из выполнения требований 1 и 2 к модели планировщика ядра следует, что для любого окна раздела планировщик ядра оповещает планировщик раздела, соответствующего данному окну, об открытии и закрытии окна строго в моменты, определенные в расписании. Согласно ограничениям корректности, накладываемым на расписание и приведенным в разделе 1.4.1, окна разделов, привязанных к одному ядру, не пересекаются. Таким образом, из выполнения перечисленных требований к моделям компонентов следует выполнение требования 2 к модели в целом.

Требование 3. *Для любого ядра системы верно, что в каждый момент времени начать передачу данных может только работа задачи, принадлежащей активному в данный момент разделу.*

Доказательство выполнения данного требования основано на выполнении требований к реализациям базовых типов автоматов **TS** и **T**. Из выполнения требования 4 к модели функциональной задачи следует, что работа может начать передачу данных только строго в момент своего завершения. Из выполнения требования 2 к модели планировщика раздела следует, что момент

завершения работы (входящий в интервал выполнения работы) принадлежит окну этого раздела. Таким образом, из выполнения перечисленных требований к моделям компонентов следует выполнение требования 3 к модели в целом.

Требование 4. *Для любого раздела верно, что в момент закрытия окна этого раздела работа этого раздела не может быть поставлена на выполнение.*

Момент закрытия окна раздела может совпасть с моментом появления новой готовой работы или с моментом завершения некоторой работы. Согласно описанию обобщенной модели функционирования МВС (см. раздел 2.4) каналы $sleep$ имеют больший приоритет в модели МВС, чем каналы $exec$. Поэтому если в некоторый момент времени возможна как синхронизация по каналу $sleep_i$ (закрытие очередного окна i-го раздела), так и синхронизация по каналу $exec_{ij}$ (постановка на выполнение работы j-й задачи i-го раздела), то выполняется синхронизация по каналу $sleep_i$, а после этой синхронизации выполнение синхронизации по каналу $exec_{ij}$ невозможно до тех пор, пока не будет выполнена синхронизация по каналу $wakeup_i$ (в соответствии с требованием 2 к модели планировщика раздела). Окон нулевой длины в расписании быть не может, поэтому синхронизация по каналу $wakeup_i$ не может быть выполнена одновременно с синхронизацией по каналу $sleep_i$ (для одного и того же i). Таким образом, синхронизация по каналу $sleep_i$ не может быть выполнена одновременно с синхронизацией по каналу $exec_{ij}$ (для одного и того же i). Следовательно, требование 4 к модели в целом выполняется.

Требование 5. *Для любого раздела верно, что если момент закрытия окна раздела совпадает с моментом завершения выполняющейся на вычислительном ядре работы этого раздела, то сначала происходит завершение работы, а затем закрытие окна, то есть завершение работы не переносится на начало следующего окна данного раздела*.

Согласно разделу 2.4 каналы $sleep$ имеют меньший приоритет в модели МВС, чем каналы $finished$. Поэтому если в некоторый момент времени возможна как синхронизация по каналу $sleep_i$ (закрытие очередного окна i-го раздела), так и синхронизация по каналу $finished_i$ (завершение работы некоторой задачи i-го раздела), то выполняется синхронизация по каналу $finished_i$. При этом из выполнения требования 8 к модели планировщика раздела следует, что сразу после синхронизации по каналу $finished_i$ выполняется синхронизация по каналу $sleep_i$. Таким образом, требование 5 к модели в целом выполняется.

Требование 6. *Для любого раздела верно, что если в некоторый момент времени становятся готовыми несколько работ раздела, то выбор, должна ли какая-либо работа быть поставлена на выполнение в этот момент, осуществляется однозначно. При этом если некоторая работа должна быть поставлена на выполнение, то ее выбор осуществляется однозначно.*

Данное требование аналогично требованиям 4.4 и 4.5 к модели планировщика раздела. Отличие требования 6 к модели в целом от указанных требований состоит в том, что в требованиях 4.4 и 4.5 к модели планировщика раздела идет речь о появлении одной готовой работы. Требования 4.4 и 4.5 допускают, например, следующую последовательность событий, происходящих в один момент времени: появление готовой работы 1, постановка на выполнение работы 1, появление готовой работы 2, вытеснение работы 1, постановка на выполнение работы 2. В то же время, если работы 1 и 2 становятся готовыми одновременно, то в реальной вычислительной системе на выполнение сразу должна быть поставлена работа 2. Согласно разделу 2.4 каналы $ready$ имеют

больший приоритет в модели МВС, чем каналы $exec$, поэтому для любого момента времени верно, что если в этот момент времени возможны несколько синхронизаций по каналам $ready$ и $exec$, то сначала будут выполнены все синхронизации по каналам $ready$ и лишь затем — возможные синхронизации по каналам $exec$. То есть сначала планировщики разделов будут уведомлены о появлении всех новых готовых работ, и лишь затем произойдет постановка работ на выполнение. Таким образом, из выполнения требований 4.4 и 4.5 к модели планировщика раздела, а также из информации о приоритетах каналов следует выполнение требования 6 к модели в целом.

Требование 7. *Для любой работы верно, что в момент, соответствующий ее завершению, она не может быть вытеснена.*

Вообще говоря, момент завершения работы может совпасть с моментом появления новой готовой работы, которая должна быть поставлена на выполнение. В этом случае работа не должна вытесняться (что означает перенос ее завершения на следующий интервал выполнения), а должна завершаться. Выполнение требования 7 обеспечивается тем, что каналы $finished$ имеют больший приоритет в модели МВС, чем каналы $preempt$.

3.4. Детерминированность моделей МВС

В разделе 3.1 было введено определение детерминированности модели МВС: модель МВС с заданной конфигурацией является детерминированной, если ВД, построенные по всем возможным вычислениям этой сети, эквивалентны (определение эквивалентности ВД дано в разделе 2.3). Это определение детерминированности модели не накладывает ограничений на значения переменных и внутреннюю логику моделей, так как целью моделирования в рамках данной работы является проверка выполнения ограничений реального времени, осуществляемая посредством анализа ВД. Так как, согласно предположениям, введенным в разделе 1.4.2, ВД МВС с заданной конфигурацией определяется однозначно, то и ВД модели МВС с заданной конфигурацией должна определяться однозначно.

Докажем детерминированность всех моделей МВС, формируемых согласно предложенному в работе методу. В доказательстве предполагается, что все компоненты модели удовлетворяют соответствующим требованиям раздела 3.2, а все входные данные — корректны (ограничения на входные данные приведены в разделе 1.4.1).

Докажем детерминированность моделей методом от противного. Предположим, что некоторая модель, построенная согласно предложенному методу, не является детерминированной, то есть для экземпляра сети автоматов, соответствующей некоторой конфигурации, могут быть получены две не эквивалентные ВД. Упорядочим события этих двух ВД по времени. Так как события в системе могут происходить одновременно, то в каждой ВД одному значению модельного времени соответствует некоторое множество событий. Пусть t^* — наименьшее значение модельного времени, которому в двух ВД соответствуют различные множества событий. Это означает, что как минимум одно событие синхронизации $e = \langle CH, A^s, t* \rangle$ присутствует в множестве событий для

этого значения времени одной ВД (будем считать эту ВД первой) и отсутствует в соответствующем множестве другой ВД (будем считать эту ВД второй) (*).

Рассмотрим все возможные варианты каналов CH события синхронизации e. Покажем, что для каждого канала предположение не верно, то есть если событие синхронизации по каналу присутствует в первой ВД, то оно присутствует и во второй ВД (для краткости далее под «событием CH» будем понимать событие синхронизации по каналу CH).

1. $wakeup_j$ или $sleep_j$. Следовательно, $A^s = \{CS_i, TS_j\}$, где CS_i — модель планировщика i-го ядра, TS_j — модель планировщика j-го раздела, привязанного к i-му ядру. Из выполнения требования 8 к моделям планировщиков раздела и требований 1 и 2 к моделям планировщиков ядра следует, что полученные ВД соответствуют конфигурациям, различающимся расписаниями окон для ядра j-го раздела. Это противоречит тому, что ВД получены с использованием одного и того же экземпляра модели. Следовательно, данный вариант не возможен.
2. $finished_j$. Следовательно, $A^s = \{TS_j, T_{jk}\}$, где TS_j — модель планировщика j-го раздела, T_{jk} — модель k-й задачи j-го раздела. Возможны два варианта:
   - Согласно первой ВД некоторая работа k-й задачи j-го раздела выполнялась на вычислительном ядре ровно WCET единиц времени. До момента t^* ВД совпадают, поэтому из отсутствия во второй ВД события $finished_j$ следует, что согласно второй ВД эта работа выполнялась на вычислительном ядре более WCET единиц времени, что противоречит требованию 3 к модели задачи; следовательно, данный вариант не возможен.
   - Согласно первой ВД для некоторой работы k-й задачи j-го в момент t^* наступила правая граница директивного интервала и эта работа выполнялась на вычислительном ядре менее WCET единиц времени. Значение этой границы зависит только от номера работы и характеристик соответствующей задачи, поэтому и согласно второй ВД момент t^* является правой границей директивного интервала той же работы; однако, согласно второй ВД, в момент t^* эта работа не завершается, что противоречит требованию 3 к модели задачи; следовательно, данный вариант не возможен.
3. $send_{jk}$. Следовательно, $A^s = \{T_{jk}, L_{h_1}, ..., L_{h_m}\}$, где T_{jk} — модель k-й задачи j-го раздела, $L_{h_1}, ..., L_{h_m}$ — модели виртуальных каналов, по которым передаются выходные сообщения этой задачи. Из выполнения требования 4 к модели задачи следует, что в первой ВД в момент t^* присутствует событие $finished_j$ и t^* не превышает правой границы директивного интервала завершившейся работы. Из рассуждений предыдущего пункта следует, что событие $finished_j$ присутствует и во второй ВД с такой же временной меткой. Следовательно, согласно второй ВД, нарушено требование 4 к модели задачи, поэтому данный вариант не возможен.
4. $receive_{jk}$. Следовательно, $A^s = \{T_{jk}, L_{h_1}, ..., L_{h_m}\}$, где T_{jk} — модель k-й задачи j-го раздела, $L_{h_1}, ..., L_{h_m}$ — модели виртуальных каналов, по которым передаются входные сообщения этой задачи. Из наличия данного события в первой ВД и выполнения требований 1 и 2 к модели ВК следует, что в первой ВД присутствует событие $send_{j_1k_1}$ с

временной меткой $(t^* - d)$, где d — задержка на передачу сообщения через виртуальный канал между k_1-й задачей j_1-го раздела и k-й задачей j-го раздела. Возможны два варианта:

- $d > 0$. В этом варианте событие $send_{j_1k_1}$ присутствует и во второй ВД, так как до момента t^* ВД совпадают; отсутствие события $receive_{jk}$ в момент t^* в таком случае говорит о невыполнении требований 1 либо 2 к модели ВК, поэтому данный вариант невозможен.
- $d = 0$. Если событие $send_{j_1k_1}$ присутствует и во второй ВД в момент t^*, то это свидетельствует о невыполнении требований 1 либо 2 к модели ВК (аналогично варианту $d > 0$), что невозможно; следовательно, событие $send_{j_1k_1}$ не присутствует во второй ВД, но это противоречит рассуждениям пункта 3 и поэтому невозможно.

5. $ready_j$. Следовательно, $A^s = \{TS_j, T_{jk}\}$, где TS_j — модель планировщика j-го раздела, T_{jk} — модель k-й задачи j-го раздела. Событие соответствует тому, что некоторая работа k-й задачи j-го раздела стала готовой к выполнению согласно первой ВД. Из выполнения требования 2 к модели задачи следует, что t^* не меньше левой границы директивного интервала работы. Возможны два варианта:
   - k-я задача j-го раздела не имеет зависимостей по данным. В этом случае отсутствие во второй ВД в момент t^* события $ready_j$ свидетельствует о нарушении требования 2 к модели задачи и поэтому невозможно.
   - k-я задача j-го раздела имеет m входных сообщений, которым соответствуют переменные $is\_data\_ready_1, ..., is\_data\_ready_m$. Согласно вспомогательному требованию к требованию 2 к модели задачи в начале периода, соответствующего моменту t^*, все эти переменные равны 0. Из совпадения ВД до момента t^*, выполнения требования 2 к модели задачи и требования 1 к модели ВК (переменная $is\_data\_ready_i$ должна принять значение 1 ровно через d_i единиц времени после синхронизации по соответствующему каналу $send$, где d_i задано в описании конфигурации МВС) следует, что все переменные $is\_data\_ready_1, ..., is\_data\_ready_m$ равны 1 в момент t^*, согласно обеим ВД. Левая граница директивного интервала рассматриваемой работы одна и та же для обеих ВД. Таким образом, отсутствие во второй ВД в момент t^* события $ready_j$ свидетельствует о нарушении требования 2 к модели задачи и поэтому невозможно.

6. $exec_{jk}$. Следовательно, $A^s = \{TS_j, T_{jk}\}$, где TS_j — модель планировщика j-го раздела, T_{jk} — модель k-й задачи j-го раздела. Событие соответствует тому, что некоторая работа k-й задачи j-го раздела поставлена на выполнение. Из выполнения требования 4 к модели МВС в целом следует, что t^* не является моментом закрытия окна раздела. Из выполнения требования 3 к модели планировщика раздела и требования 5 к модели задачи следует, что данная работа готова и t^* принадлежит ее директивному интервалу. Из совпадения ВД до момента t^* и из рассуждений пункта 5 следует, что согласно обеим ВД на момент t^* есть

как минимум одна готовая работа. Из выполнения требования 4.1 к модели планировщика раздела следует, что имеет место хотя бы один из трех вариантов:

- В первой ВД в момент t^* присутствует событие $wakeup_j$. Из рассуждений пункта 1 следует, что во второй ВД также присутствует это событие с такой же временной меткой. Из выполнения требований 4.2 и 4.5 к модели планировщика раздела следует, что в момент t^* во второй ВД присутствует событие $exec_{jk}$, что противоречит предположению (*).
- В первой ВД в момент t^* присутствует событие $finished_j$. Из рассуждений пункта 2 следует, что во второй ВД также присутствует это событие с такой же временной меткой. Из выполнения требований 4.3 и 4.5 к модели планировщика раздела следует, что в момент t^* во второй ВД присутствует событие $exec_{jk}$, что противоречит предположению (*).
- В первой ВД в момент t^* присутствует событие $ready_j$. Из рассуждений пункта 5 следует, что во второй ВД также присутствует это событие с такой же временной меткой. Из выполнения требований 4.4 и 4.5 к модели планировщика раздела следует, что в момент t^* во второй ВД присутствует событие $exec_{jk}$, что противоречит предположению (*).

7. $preempt_{jk}$. Событие соответствует вытеснению с вычислительного ядра некоторой работы. Согласно требованиям 5, 6 и 7 к модели планировщика раздела имеет место один из двух вариантов:
   - В первой ВД в момент t^* присутствует событие $sleep_j$. Согласно рассуждениям пункта 1, во второй ВД также присутствует событие $sleep_j$ с той же временной меткой. Из выполнения требования 4 к модели МВС в целом следует, что событие $exec_{jk}$, предшествующее $preempt_{jk}$, имеет временную метку, строго меньшую t^*. Поэтому событие $exec_{jk}$ присутствует и во второй ВД. Следовательно, согласно требованию 7 к модели планировщика раздела, в момент t^* во второй ВД имеется либо событие $preempt_{jk}$, что противоречит предположению (*), либо событие $finished_j$, что противоречит требованию 3 к модели задачи.
   - В первой ВД в момент t_i присутствует событие $ready_j$. Согласно рассуждениям пункта 5, это же событие присутствует и во второй ВД. Из выполнения требований 4.1 и 4.4 к модели планировщика раздела следует, что во второй ВД в момент t^* присутствует событие $exec_{j_1k_1}$. Это же событие присутствует и в первой ВД (см. пункт 6). Из совпадения ВД до момента t^* следует, что согласно обеим ВД непосредственно перед моментом t^* на данном вычислительном ядре выполняется одна и та же работа. Поэтому из выполнения требования 6 к модели планировщика следует, что во второй ВД, как и в первой, должно присутствовать событие $preempt_{jk}$, что противоречит предположению (*).

Рассмотрим отдельно случай, когда $t^* = 0$. Поскольку согласно разделу 1.4.1 длительности окон и выполнения работ не могут быть равны 0, то в момент $t^* = 0$ возможны синхронизации по следующим каналам:

- $wakeup_j$. Противоречие предположению (*) доказывается аналогично п. 1 выше.

– $ready_j$. Противоречие предположению (*) доказывается аналогично п. 5 выше (см. вариант отсутствия зависимостей по данным, т.к. в момент $t^* = 0$ ни одна из работ задач-отправителей не успевает завершиться).
– $exec_{jk}$. Противоречие предположению (*) доказывается аналогично п. 6 выше (см. варианты наличия в момент t^* событий $wakeup_j$ и $ready_j$, т.к. в момент $t^* = 0$ ни одна из работ задач-отправителей не успевает завершиться).

Таким образом, доказано от противного, что предположение о существовании двух различных ВД, соответствующих одному экземпляру рассматриваемой сети автоматов, ложно. Следовательно, соответствующая модель МВС является детерминированной.

Глава 4. Инструментальная система проверки выполнения ограничений реального времени для конфигураций МВС

При работе над данной главой диссертации использованы следующие публикации автора, в которых, согласно Положению о присуждении ученых степеней в МГУ, отражены основные результаты, положения и выводы исследования:

Глонина А. Б. Инструментальная система проверки выполнения ограничений реального времени для конфигураций модульных вычислительных систем // Вестник Московского университета. Серия 15: Вычислительная математика и кибернетика. — 2020. — № 3. — С. 16–29.

К указанным публикациям, согласно Положению о присуждении ученых степеней в МГУ, может быть приравнено, по решению диссертационного совета, свидетельство о регистрации программы для ЭВМ:

Глонина А. Б. Инструментальная система проверки выполнения ограничений реального времени для конфигураций модульных вычислительных систем. ФИПС. Св. о рег. программы для ЭВМ № 2020611082 от 23.01.2020.

4.1. Требования к инструментальной системе

В разделе 1.3 сформулированы требования, которым должны удовлетворять инструментальные средства имитационного моделирования МВС. Требования 1—3 относятся к математическому аппарату, лежащему в основе средств, а требования 4—7 — к программной реализации. В главе 2 была предложена обобщенная модель функционирования МВС, удовлетворяющая требованиям 1—3. Перечислим требования, которым должна удовлетворять инструментальная система проверки выполнения ограничений реального времени для МВС, в которой реализован предложенный подход к построению модели МВС с заданной конфигурацией.

4. *Поддержка библиотеки моделей компонентов МВС.*
5. *Автоматизация построения и прогона модели.*
6. *Возможность включения в модель МВС и в библиотеку моделей пользовательских моделей компонентов МВС.*
7. *Скорость* построения и прогона модели, а также анализа результатов моделирования должна позволять работать с конфигурациями размерности, соответствующей реальным системам, с учетом числа конфигураций, оцениваемых при проектировании (это число может достигать нескольких сотен и более при работе оптимизационных алгоритмов).

Описанная в настоящем разделе инструментальная система была разработана автором с выполнением перечисленных требований. Исходный код доступен по адресу:

`https://github.com/AlevtinaGlonina/MCSSim`.

4.2. Состав и схема работы инструментальной системы

В таблице 4.1 приведен состав разработанной инструментальной системы проверки выполнения ограничений реального времени для МВС.

1. **UPPAAL** — инструмент построения и верификации временных автоматов с остановкой таймеров, разработанный в университете Уппсалы [90]. В рамках настоящей работы графический редактор UPPAAL используется для создания, модификации и просмотра автоматов (как автоматов-моделей компонентов МВС, так и автоматов-наблюдателей для проверки выполнения требований корректности к моделям). Верификатор UPPAAL используется для проверки выполнения требований корректности к моделям компонентов МВС. Согласно разделу 3.2, проверка каждого требования выполняется посредством проверки недостижимости «плохой» локации в сети автоматов, состоящей из автомата-модели и автомата-наблюдателя.

2. **Автоматная модель компонента МВС** — параметризованный автомат, моделирующий функционирование компонента МВС и представленный в формате средства UPPAAL.

3. **Требования корректности к моделям компонентов МВС** представлены в виде **набора автоматов-наблюдателей**, соответствующих требованиям и описанных в формате средства UPPAAL. В разделах 3.2.2—3.2.2 для каждого базового типа автоматов сформулированы требования корректности, которым должны удовлетворять все параметризованные автоматы, реализующие этот базовый тип. Разработанные автоматы-наблюдатели, соответствующие этим требованиям, описаны в разделах В.1—В.4 приложения В. Помимо требований, соответствующих базовым типам автоматов, в разделе 3.2.2 автором были сформулированы требования, специфичные для построенных моделей конкретных типов компонентов МВС. Автоматы-наблюдатели для проверки этих требований описаны в разделе В.5 приложения В. При разработке новых моделей конкретных типов компонентов МВС для этих моделей могут быть сформулированы соответствующие наборы требований и созданы автоматы-наблюдатели для проверки выполнения требований.

4. **Библиотека автоматных моделей компонентов МВС** — набор автоматных моделей компонентов МВС; для каждой модели этого набора выполнены все соответствующие требования корректности. Данная библиотека содержит разработанные автором модели планировщика ядра, функциональной задачи, виртуального канала, а также трех часто используемых в МВС планировщиков работ раздела: с фиксированными приоритетами и вытеснением; с фиксированным приоритетами без вытеснения; планировщик, работающий по стратегии EDF с вытеснением. Логика работы этих моделей описана в разделе 2.7, а подробное описание приведено в приложении А. Библиотека может быть пополнена новыми моделями. Для этого необходимо проверить с использованием верификатора UPPAAL, что для новых моделей выполнены соответствующие требования (как минимум — требования для базовых типов компонентов МВС).

5. **Библиотека моделирования сетей временных автоматов** — разработанная автором библиотека на языке C++, содержащая программные реализации всех элементов математического аппарата сетей временных автоматов с остановкой таймеров и используемая для получения вычисления сети автоматов, моделирующей МВС. В библиотеке имеются классы, соответствующие

Таблица 4.1 — Состав инструментальной системы проверки выполнения ограничений реального времени для МВС

Компонент инструментальной системы или создаваемый объект	Краткое описание компонента/объекта
UPPAAL	Средство построения и верификации временных автоматов.
Автоматная модель компонента МВС	Параметризованный временной автомат, моделирующий функционирование компонента МВС.
Требования корректности к моделям компонентов МВС	Набор автоматов-наблюдателей, соответствующих требованиям корректности к компоненту МВС.
Библиотека автоматных моделей компонентов МВС	Набор автоматов, моделирующих стандартные компоненты МВС, и удовлетворяющих соответствующим требованиям корректности.
Библиотека моделирования сетей временных автоматов	Библиотека, содержащая программные реализации всех элементов математического аппарата сетей временных автоматов с остановкой таймеров.
Программная модель компонента МВС	Код на языке C++, моделирующий функционирование компонента МВС и полученный с помощью генератора кода по автоматной модели этого компонента МВС.
Генератор кода	Средство, позволяющее сгенерировать для параметризованного автомата, описанного в формате UPPAAL, его представление на языке C++.
Библиотека программных моделей компонентов МВС	Библиотека, полученная с помощью генератора кода по автоматам из библиотеки автоматных моделей компонентов МВС.
Компилятор и редактор связей	Стандартные инструменты для получения исполняемого кода программы из ее исходных текстов с использованием библиотек.
Программа построения и запуска модели МВС (исходный текст)	Текст программы, реализующей параметризованную модель МВС. Программа принимает на вход описание конфигурации конкретной МВС. Результатом ее работы является ВД функционирования этой МВС.
Параметризованная программная модель МВС	Исполняемый файл, полученный в результате компиляции и сборки исходного кода программы построения и запуска модели МВС.
Файл описания конфигурации МВС	Файл, содержащий информацию о количестве, типах, числовых характеристиках и связях компонентов системы
Файл с ВД МВС	Файл с данными о событиях постановки на выполнение, вытеснения и завершения работ
Анализатор ВД МВС	Программа, проверяющая выполнение ограничений реального времени для МВС с заданной конфигурацией посредством анализа ВД этой МВС.

автомату, локации, переходу, таймеру и т.д. В открытых источниках не было найдено средства, позволяющего автоматически получать вычисления сетей автоматов с остановкой таймеров. UPPAAL предоставляет для этого лишь графический интерфейс, требующих ручных действий, что не соответствует требованию 5 раздела 4.1. Поэтому было принято решение о разработке собственного средства моделирования сетей временных автоматов. Более подробно библиотека описана в разделе 4.3.

6. **Программная модель компонента МВС** — код на языке C++, моделирующий функционирование компонента МВС и являющийся C++-представлением параметризованного автомата-модели этого компонента. В данном коде используются классы, реализованные в библиотеке моделирования сетей временных автоматов. Программная модель компонента МВС формируется автоматически по соответствующей автоматной модели с использованием генератора кода.

7. **Генератор кода** — разработанный автором инструмент, позволяющий сгенерировать для параметризованного автомата, описанного в формате UPPAAL, его представление на языке C++. Генератор необходим для автоматического получения по автоматным моделям компонентов МВС соответствующих программных моделей, так как выполнение такой трансляции вручную требует существенных трудозатрат и, кроме того, может привести к внесению ошибок в модели. Данный инструмент написан на языке Python, с использованием средства генерации текста по шаблонам Cheetah [132]. Сгенерированный код содержит объекты классов библиотеки моделирования сетей временных автоматов, а также классы-наследники библиотечных классов и объекты этих классов-наследников. Кроме того, в сгенерированном коде имеются функции, каждая из которых соответствует одному базовому типу автоматов. Такая функция принимает на вход имя параметризованного автомата, реализующего базовый тип, и создает объект класса, соответствующего этому параметризованному автомату. Наличие таких функций дает возможность не модифицировать код построения экземпляра модели МВС по описанию конфигурации при появлении новых моделей планировщиков разделов: для порождения объекта соответствующего класса достаточно указать имя модели планировщика в описании конфигурации.

8. **Библиотека программных моделей компонентов МВС** — библиотека на языке C++, полученная автором с помощью генератора кода по автоматам из библиотеки автоматных моделей компонентов МВС. Для того, чтобы добавить в библиотеку новую модель, необходимо добавить соответствующий параметризованный автомат, реализующий один из базовых типов автоматов, в библиотеку автоматных моделей, а затем запустить для обновленной библиотеки автоматных моделей генератор кода.

9. **Компилятор и редактор связей** — стандартные инструменты из набора GCC [133] для получения исполняемого кода программы из ее исходных текстов с использованием библиотек. Необходимы для получения параметризованной программной модели МВС, использующей библиотеку моделирования сетей временных автоматов и библиотеку программных моделей компонентов МВС.

10. **Программа построения и запуска модели МВС (исходный текст)** — разработанная автором программа на языке C++, реализующая последовательное выполнение трех этапов:

a) Построение модели МВС по описанию конфигурации системы в соответствии с методом, предложенным в разделе 2.5.

b) Прогон построенной модели, в процессе которого формируется ВД сети автоматов.

c) Построение по ВД сети автоматов искомой ВД МВС в соответствии с алгоритмом, приведенном в разделе 2.6.

11. **Параметризованная программная модель МВС** — исполняемый файл, получаемый в результате компиляции и сборки исходного кода программы построения и запуска модели МВС с использованием библиотеки программных моделей компонентов МВС и библиотеки моделирования сетей временных автоматов.

12. **Файл описания конфигурации МВС** — файл в формате XML, содержащий информацию о количестве, типах, числовых характеристиках и связях компонентов системы. Используется параметризованной программной моделью МВС для построения модели конкретной МВС, а также анализатором ВД для проверки выполнения ограничений реального времени для конфигурации МВС.

13. **Файл с ВД МВС** — файл в формате XML с данными о событиях постановки на выполнение, вытеснения и завершения работ. Используется анализатором ВД для проверки выполнения ограничений реального времени для конфигурации МВС.

14. **Анализатор ВД МВС** — разработанная автором программа на языке C++, проверяющая выполнение ограничений реального времени для МВС с заданной конфигурацией в соответствии с условием RT, сформулированным в разделе 1.4.3. Результатом работы программы является булево значение *True* или *False*.

Схема получения параметризованной программной модели МВС приведена на рисунке 4.1.

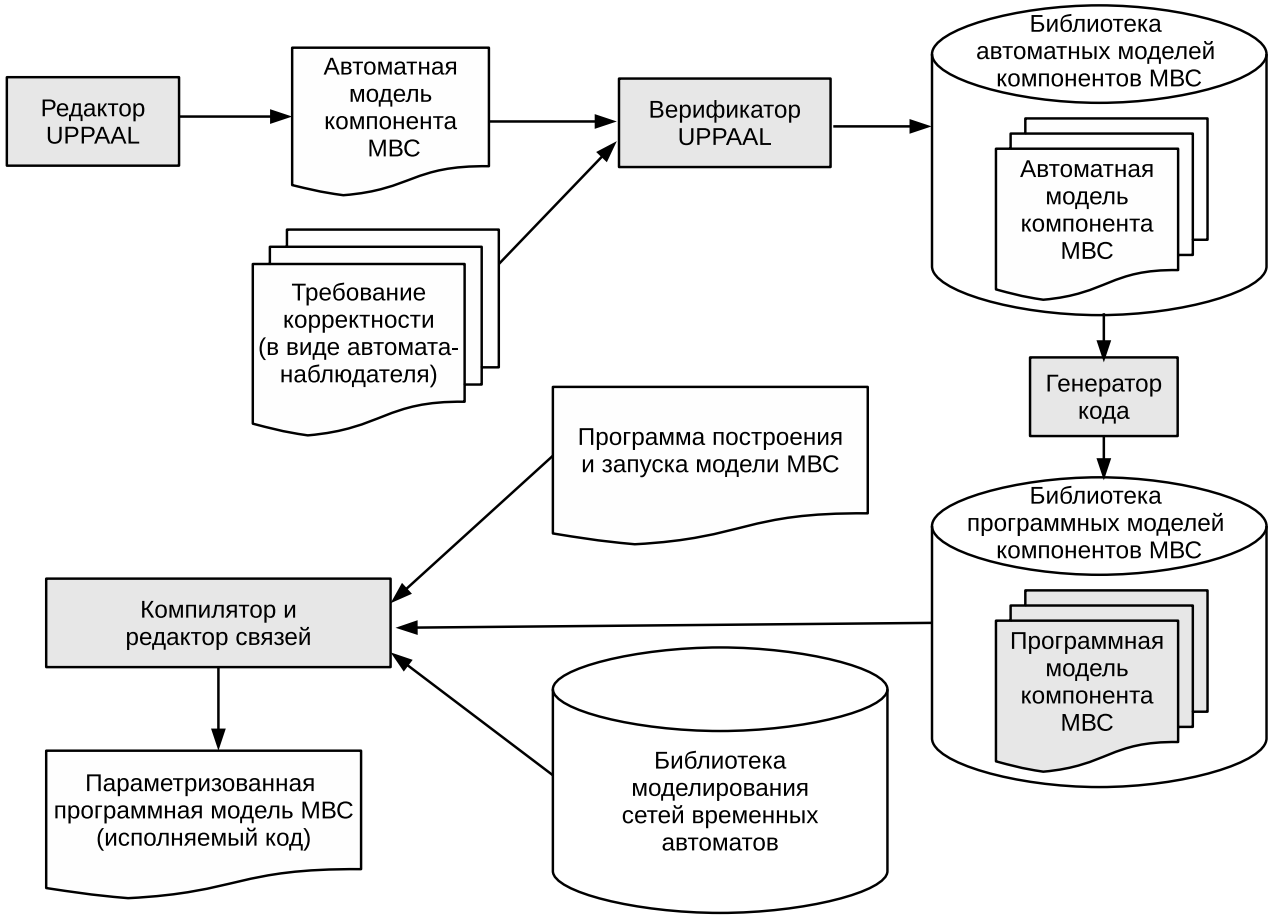

Рисунок 4.1 — Схема получения параметризованной программной модели МВС

Автоматные модели компонентов МВС создаются в редакторе UPPAAL [90]. Далее с помощью верификатора UPPAAL для каждой разработанной модели проверяется выполнение требований корректности, сформулированных для соответствующего базового типа автоматов, а также, возможно, дополнительных требований, специфичных для этой модели. Таким образом, выполняется требование 4 раздела 4.1.

Автоматные модели компонентов МВС, для которых выполнены все необходимые требования, формируют библиотеку автоматных моделей компонентов МВС. Для каждой автоматной модели с помощью генератора кода формируется соответствующая ей программная модель. Полученные программные модели формируют библиотеку программных моделей компонентов МВС.

Далее посредством компиляции и сборки на основе библиотеки программных моделей компонентов МВС, библиотеки моделирования сетей временных автоматов и исходного кода программы построения и запуска модели МВС получается исполняемая параметризованная программная модель МВС.

Важной особенностью разработанной инструментальной системы является то, что после добавления в библиотеку автоматных моделей новой модели компонента МВС для использования этой модели достаточно запуска генератора кода и пересборки исполняемой параметризованной программной модели МВС. Модификации кода этой модели не требуется. Следовательно, выполняется требование 6 раздела 4.1. Для фиксированного состава библиотеки программных моделей компонентов МВС моделирование функционирования МВС различных конфигураций не требует пересборки исполняемой параметризованной программной модели, что позволяет сократить временные затраты при анализе большого числа конфигураций.

Схема использования параметризованной программной модели МВС для проверки выполнения ограничений реального времени приведена на рисунке 4.2.

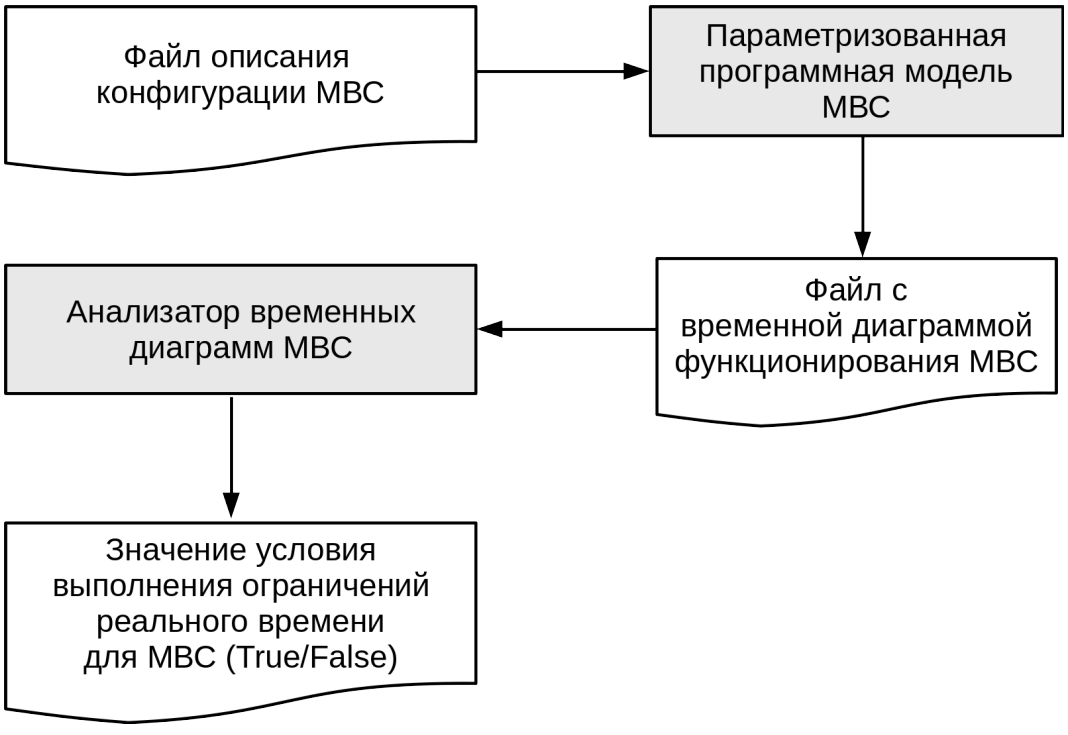

Рисунок 4.2 — Схема использования параметризованной программной модели МВС

Для проверки выполнения ограничений реального времени для некоторой конфигурации МВС необходимо сформировать файл в формате XML с описанием этой конфигурации. Этот файл подается на вход параметризованной программной модели МВС. Результатом прогона этой модели является файл в формате XML, содержащий описание ВД МВС. Далее ВД МВС обрабатывается Анализатором ВД МВС, который проверяет выполнение условия RT, сформулированного в разделе 1.4.3. Этап проверки выполнения ограничений реального времени для МВС может выполняться другими, отличными от разработанного автором Анализатора, средствами. Например, при интеграции с САПР «Планировщик ИМА» (см. раздел 4.4), проверка выполнения ограничений реального времени для МВС выполняется средствами самой САПР. В соответствии с описанной схемой, построение и прогон модели МВС с заданной конфигурацией выполняется автоматически и, следовательно, выполняется требование 5 раздела 4.1.

Таким образом, разработанное программное средство удовлетворяет требованиям 4—6 раздела 4.1. Выполнение требования 7 проверено экспериментально (результаты приведены в главе 5).

4.3. Библиотека моделирования сетей временных автоматов с остановкой таймеров

В открытых источниках не было найдено средства моделирования сетей временных автоматов с остановкой таймеров, позволяющего автоматически получать вычисления или временные диаграммы для таких сетей, поэтому было принято решение разработать собственное средство моделирования таких сетей временных автоматов.

Автором была разработана на языке C++ библиотека моделирования сетей временных автоматов, диаграмма классов которой приведена на рисунке 4.3.

Библиотека содержит следующие классы:

- класс *Network* соответствует сети временных автоматов с остановкой таймеров. Содержит набор автоматов, каналов, переменных и таймеров, в т.ч. служебный таймер для хранения значения модельного времени.
- класс *Automaton* соответствует одному автомату. Содержит набор локаций и хранит набор событий синхронизации, выполненных автоматом в процессе его функционирования в составе сети автоматов.
- класс *Channel* соответствует каналу синхронизации. Класс предоставляет интерфейс для отправки и получения сигнала по каналу.
- класс *DuplexChannel* является наследником класса *Channel* и соответствует каналу «точка-точка».
- класс *BroadcastChannel* является наследником класса *Channel* и соответствует широковещательному каналу.
- класс *Var* соответствует целочисленной переменной.

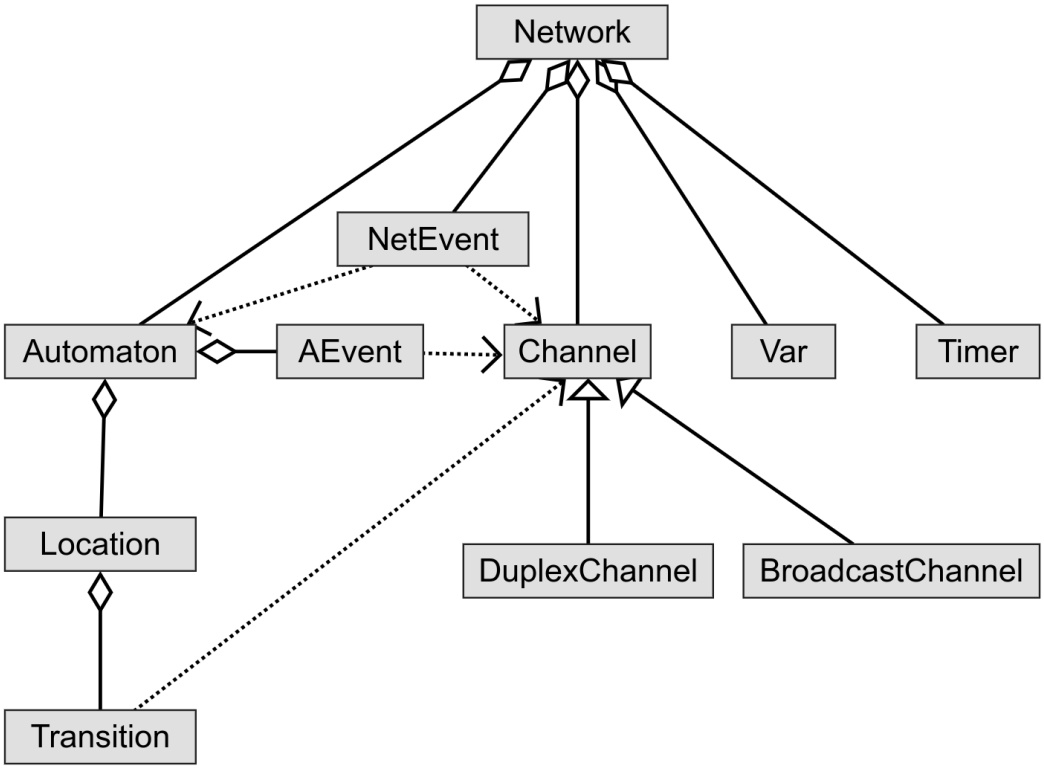

Рисунок 4.3 — Диаграмма классов библиотеки моделирования сетей временных автоматов с остановкой таймеров

– класс *Timer* соответствует таймеру.
– класс *Location* соответствует локации автомата. Содержит список переходов, которые могут быть выполнены из этой локации, и имеет виртуальные функции, соответствующие инварианту и условиям активности таймеров для локации.
– класс *Transition* соответствует переходу в автомате. Имеет виртуальные методы, соответствующие условию, действиям по изменению переменных и таймеров, и действиям синхронизации. Также содержит указатель на локацию, в которую осуществляется переход.
– класс *AEvent* соответствует событию синхронизации автомата и содержит информацию об этом событии, в т.ч. канал, с использованием которого выполняется синхронизация, и модельное время синхронизации.
– класс *NetEvent* соответствует событию синхронизации в сети автоматов и содержит информацию об этом событии: канал, список автоматов, участвующих в синхронизации, и модельное время синхронизации.

Перечисленные классы реализуют функционирование соответствующих объектов математического аппарата сетей временных автоматов с остановкой таймеров согласно описаниям в разделах 2.2 и 2.3.

Библиотека имеет два класса, соответствующих событию синхронизации, по следующей причине. Согласно разделу 2.6, для построения ВД МВС по ВД сети автоматов используются лишь события синхронизации с участием моделей функциональных задач. Для ускорения построения ВД МВС удобно на этапе построения вычисления сети автоматов для каждого автомата хранить

набор событий синхронизации, в которых он участвует. Тогда при построении ВД МВС достаточно рассмотреть лишь наборы событий автоматов-моделей функциональных задач (т.е. наборы объектов *AEvent*), а не выполнять для каждого объекта *NetEvent* поиск таких автоматов в списке участников синхронизации. В то же время, разработанная библиотека может использоваться для задач, отличных от моделирования МВС, поэтому она также позволяет получить ВД сети автоматов в общем виде (т.е. в виде набора объектов *NetEvent*). Указанные представления ВД сети автоматов (набор объектов *NetEvent* и совокупность наборов объектов *AEvent*) по построению эквивалентны, то есть соответствуют одной и той же ВД сети автоматов, и могут быть получены одно из другого.

Средствами разработанной библиотеки можно получить одну возможную ВД заданной сети автоматов. Если для сети автоматов существует несколько возможных ВД, то будет построена одна из возможных ВД. В разделе 3.4 было доказано, что сети автоматов, построенные на основе предложенной обобщенной модели МВС, детерминированы. То есть для каждой такой сети автоматов верно, что ВД, построенные по всем возможным вычислениям этой сети, эквивалентны. Поэтому функциональности разработанной библиотеки достаточно для использования при решении задачи данной работы.

4.4. Интеграция с САПР «Планировщик ИМА»

Для апробации разработанной инструментальной системы была выполнена ее интеграция с САПР «Планировщик ИМА» [19, 27].

САПР «Планировщик ИМА» используется при проектировании МВС с архитектурой ИМА и предназначена для решения задачи распределения рабочей нагрузки по вычислительным ядрам, задачи построения окон разделов и назначения приоритетов функциональных задач.

На вход САПР подается описание МВС с архитектурой ИМА, содержащее следующие характеристики:

- набор модулей;
- набор типов используемых процессоров;
- для каждого модуля — число и тип процессоров;
- для каждого процессора — число ядер, время инициализации окна и переключения контекста между разделами;
- для каждого ядра — наибольшее допустимое значение загрузки.

Также задается описание рабочей нагрузки:

- набор задач; для каждой задачи — период, начальное значение приоритета (начальные значения приоритетов нескольких задач одного раздела могут совпадать), время выполнения на каждом из типов процессоров;
- набор разделов; для каждого раздела — набор принадлежащих ему задач и набор ядер, к которым может быть привязан данный раздел;

– набор сообщений; для каждого сообщения — задача-отправитель, задача-получатель, размер и максимальные задержки на передачу через бортовую сеть и через память модуля.

Выполнением работ разделов управляет планировщик с фиксированными приоритетами и вытеснением. Считается, что левая граница директивного интервала очередной работы задачи равна началу соответствующего периода, а правая граница директивного интервала — окончанию соответствующего периода.

Необходимо найти привязку разделов к ядрам, расписание окон разделов для каждого ядра и уточненные значения приоритетов задач, уникальные в рамках раздела. При этом объем сообщений, передаваемых через бортовую сеть в течение интервала планирования, должен быть минимальным, и найденная конфигурация должна удовлетворять условию выполнения ограничений реального времени, сформулированному в разделе 1.4.3.

Для решения описанной задачи в САПР «Планировщик ИМА» используются различные оптимизационные алгоритмы (жадный [19], генетический [134], метод ветвей и границ [135]). В процессе работы алгоритма оптимизации формируется некоторая конфигурация, для которой требуется проверить выполнение ограничений реального времени. Если ограничения реального времени выполнены, конфигурация может быть принята в качестве решения задачи. В противном случае конфигурация отбрасывается и поиск продолжается.

В САПР «Планировщик ИМА» существует встроенная имитационная модель для проверки выполнения ограничений реального времени. Однако для нее не имеется формального обоснования корректности, и в ее основе нет математического аппарата, который позволил бы такое обоснование выполнить. Поэтому целесообразно заменить ее на параметризованную программную модель МВС, формируемую с помощью разработанной автором инструментальной системы и не обладающую указанными недостатками.

Схема интеграции разработанной инструментальной системы с САПР «Планировщик ИМА» представлена на рисунке 4.4. Из состава инструментальной системы на данной схеме приведена лишь параметризованная программная модель МВС, а также файлы с ее входными и выходными данными. Остальные компоненты инструментальной системы требуются для формирования этой модели и в цикле работы САПР «Планировщик ИМА» не используются.

На этапе работы поискового алгоритма, требующем проверки выполнения ограничений реального времени для некоторой конфигурации, генерируется файл описания этой конфигурации. Затем запускается параметризованная программная модель МВС, принимающая на вход сгенерированный файл. Результатом работы модели является файл с ВД МВС. Этот файл разбирается в САПР «Планировщик ИМА», и на основе его данных осуществляется проверка выполнения ограничений реального времени.

Для реализации этой схемы в САПР «Планировщик ИМА» были добавлены функции генерации файлов описаний конфигураций и разбора файлов с ВД МВС.

В отличие от общей схемы использования параметризованной программной модели МВС (см. рисунок 4.2), при интеграции с САПР «Планировщик ИМА» не используется Анализатор ВД, а его функцию выполняет непосредственно САПР. Причиной этого является то, что САПР предоставляет пользователю не только информацию о выполнении ограничений реального времени для МВС с заданной конфигурацией, но и дополнительные данные о функционировании

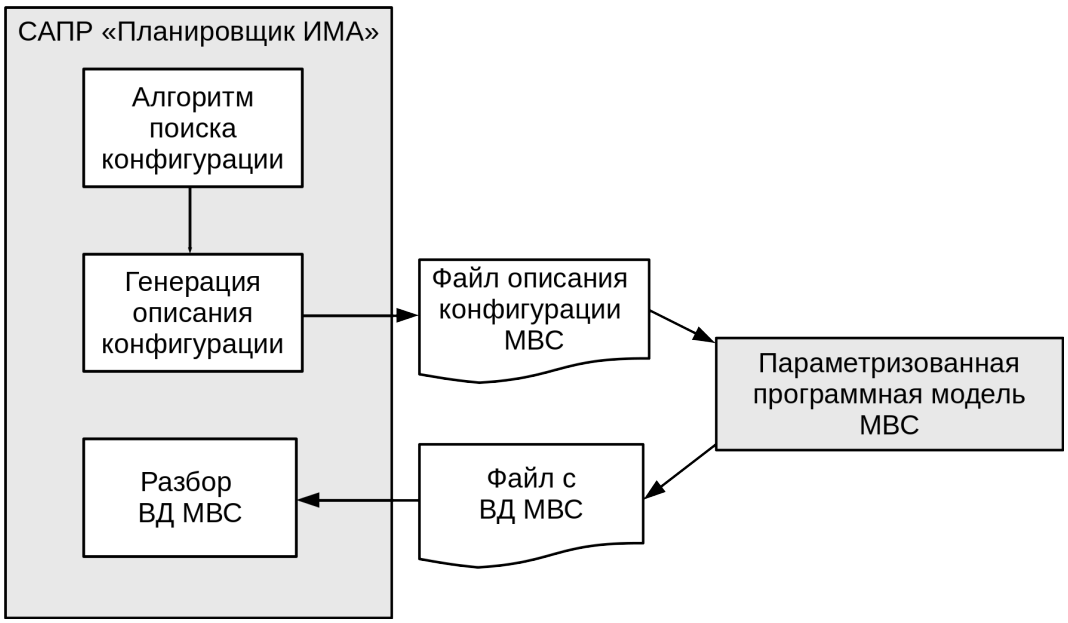

Рисунок 4.4 — Схема интеграции разработанной инструментальной системы с САПР «Планировщик ИМА»

этой МВС: загрузка ядер, информация об опоздавших работах, графическое представление ВД. Для получения этих данных необходима доступность всей ВД МВС в САПР.

Реализованная схема интеграции была успешно апробирована на данных, соответствующих одной из современных авиационных бортовых систем с архитектурой ИМА [19]. Результаты апробации приведены в разделе 5.2.3.

Глава 5. Экспериментальное исследование разработанных методов и средств

При работе над данной главой диссертации использованы следующие публикации автора, в которых, согласно Положению о присуждении ученых степеней в МГУ, отражены основные результаты, положения и выводы исследования:

Глонина А. Б. Инструментальная система проверки выполнения ограничений реального времени для конфигураций модульных вычислительных систем // Вестник Московского университета. Серия 15: Вычислительная математика и кибернетика. — 2020. — № 3. — С. 16–29.

5.1. Цели и методика экспериментального исследования

Экспериментальное исследование разработанных методов и средств имело следующие цели:

- Сравнить порядок зависимости времени построения и прогона модели МВС с заданной конфигурацией от характеристик конфигурации с теоретическими оценками, полученными в разделе 2.8.
- Оценить производительность и масштабируемость по количественным характеристикам конфигурации МВС разработанных методов и средств, а также подтвердить их пригодность для анализа систем реальной размерности.
- Сравнить по производительности разработанную параметризованную модель МВС с моделью, встроенной в САПР «Планировщик ИМА» [19].

В качестве базовых исходных данных для проведения экспериментального исследования было выбрано описание МВС, соответствующее одной из современных авиационных бортовых систем с архитектурой ИМА: порядка 150 периодических задач, сгруппированных в 10 разделов, около 100 синхронных сообщений, 6 вычислительных ядер [19]. Конфигурация МВС, соответствующая этой системе и содержащая порядка 8000 окон на интервале планирования длительностью в 2 секунды (типовые значения, используемые в САПР «Планировщик ИМА»), будет далее обозначаться как $CONF_{base}$.

На основе этого описания МВС было сформировано несколько наборов данных, использование которых позволило оценить порядок зависимости времени построения и прогона модели от различных характеристик конфигурации МВС. Часть наборов данных была сгенерирована таким образом, чтобы в серии экспериментов, проводящейся для исследования зависимости времени построения и прогона разработанной параметризованной модели МВС от некоторой характеристики конфигурации, варьировалось значение этой характеристики, а остальные характеристики конфигурации оставались неизменными. Если изменение одной характеристики конфигурации МВС невозможно без изменения другой характеристики конфигурации МВС, то значения таких характеристик варьируются в серии экспериментов согласованно.

В наборы данных для экспериментального исследования зависимости времени прогона модели от той или иной характеристики конфигурации МВС были включены только те конфигурации, на которых ограничения реального времени выполняются. Это связано с тем, что для конкретных конфигураций МВС время прогона модели также зависит от выполнения ограничений реального времени для задач этой конфигурации. Так, например, если в конфигурации многие задачи имеют зависимость по данным (прямую или опосредованную) от некоторой задачи T, то в случае опоздания и принудительного завершения работы этой задачи ни одна из соответствующих работ остальных задач не будет выполнена из-за отсутствия необходимых входных данных. Следовательно, и время прогона модели для конфигурации, в которой для работ задачи T нарушены ограничения реального времени, будет существенно меньше за счет «простоя» значительной части моделей задач и виртуальных каналов, чем для конфигурации такой же размерности (с тем же количеством задач, сообщений и т.д.), в которой ограничения реального времени выполняются для работ всех задач.

Дополнительно, чтобы подтвердить зависимость времени прогона модели от выполнения ограничений реального времени, были сформированы наборы данных, включающие конфигурации, для всех работ всех задач которых ограничения реального времени не выполняются.

Для обеспечения согласованности изменений различных характеристик конфигурации МВС использовалась САПР «Планировщик ИМА»: при изменении той или иной характеристики конфигурации МВС, с помощью САПР «Планировщик ИМА» заново решалась задача распределения разделов по вычислительным ядрам, построения окон разделов и назначения приоритетов функциональных задач.

Все эксперименты проводились на машине с процессором Intel(R) Xeon(R) CPU с тактовой частотой 2.30 ГГц и оперативной памятью размером 7.2 ГБ.

Для каждой конфигурации каждого набора данных построение и прогон модели осуществлялось несколько раз, а затем полученные результаты усреднялись. Для оценки длительности прогона модели для заданной конфигурации усреднение проводилось по 5 запускам модели. При этом отклонения длительности прогона модели для конкретных запусков от среднего значения не превышали 3%. Для оценки длительности построения модели для заданной конфигурации усреднение проводилось по 40 запускам модели. При этом отклонения длительности прогона модели для конкретных запусков от среднего значения не превышали 10%. Больший разброс значений длительности построения модели, по сравнению с разбросом значений длительности прогона, объясняется тем, что построение модели занимает единицы *милли*секунд и на его длительность оказывают влияние служебные процессы в операционной системе. Длительность прогона модели занимает единицы секунд и для этой длительности вклад внешних процессов практически не заметен.

На основе полученных экспериментальных данных методом наименьших квадратов [136] строится полином, выражающий зависимость времени построения/прогона модели от той или иной характеристики конфигурации. Степень полинома выбирается исходя из оценок, выведенных в разделе 2.8. Для оценки качества полученной регрессии вычисляется значение коэффициента детерминации (R^2) и F-статистики [136]. $R^2 = \frac{\sum_{i=1}^{n}(\hat{y}_i-\overline{y})^2}{\sum_{i=1}^{n}(y_i-\overline{y})^2}$, где n — количество

полученных экспериментально пар <значение характеристики конфигурации, время построения/прогона модели> для *различных* значений этой характеристики конфигурации, y_i — значение времени построения/прогона из i-й пары, $\hat{y}_i$ — значение, полученное для характеристики конфигурации из i-й пары, с использованием построенного полинома, $\overline{y}$ — выборочное среднее времени построения/прогона модели в рассматриваемой серии экспериментов. $F = \frac{R^2}{1-R^2} \cdot \frac{n-m}{m-1}$, где m — количество коэффициентов в построенном полиноме ($m = 2$ для линейного полинома, $m = 3$ — для квадратичного и т.д.)

Для вычисления коэффициента детерминации и F-статистики используются соответствующие встроенные функции пакета LibreOffice Calc. Величина R^2 показывает, какая часть (доля) вариации длительности построения/прогона модели обусловлена вариацией соответствующей характеристики конфигурации. При $R^2 = 1$ все полученные экспериментальным образом точки лежат на линии регрессии, при $R^2 = 0$ вариация длительности построения/прогона модели полностью обусловлена воздействием неучтенных в функции регрессии переменных. Если полученное значение F-статистики больше критического значения (табличного), то регрессия считается значимой и построенный полином действительно выражает зависимость времени построения/прогона модели от той или иной характеристики конфигурации. Табличное значение F-статистики берется для уровня значимости 0.05 и зависит от m и n.

Исследование зависимости времени построения и прогона модели от количества функциональных задач. Для этого исследования был сформирован *набор 1*, содержащий 10 конфигураций МВС, различающихся количеством функциональных задач. Первая конфигурация набора — конфигурация $CONF_{base}$. i-я конфигурация получена из $CONF_{base}$ разбиением каждой задачи на i задач. В каждой производной конфигурации сумма WCET всех задач, соответствующих одной задаче в конфигурации $CONF_{base}$, равна WCET этой задачи в $CONF_{base}$. Периоды задач в каждой производной конфигурации совпадают с периодами соответствующих задач в $CONF_{base}$. При разбиении задачи на i частей только одна из получившихся задач имеет входные и выходные сообщения. Таким образом, во всех производных конфигурациях количество сообщений одинаково и равно количеству сообщений в конфигурации $CONF_{base}$. Приоритеты задач каждой конфигурации, расписание окон, а также привязка разделов к ядрам получены с помощью САПР «Планировщик ИМА».

Исследование зависимости времени построения и прогона модели от количества сообщений. Для этого исследования был сформирован *набор 2*, содержащий 10 конфигураций МВС, различающихся количеством функциональных задач и количеством синхронных сообщений. Первая конфигурация набора — $CONF_{base}$. i-я конфигурация получена из первой выполнением тех же действий, что и при генерации набора 1 с тем лишь отличием, что при разбиении задачи на i частей каждая из получившихся задач имеет входные и выходные сообщения, соответствующие сообщениям в конфигурации $CONF_{base}$. При этом, если в конфигурации $CONF_{base}$ k-я задача посылает сообщение m-й задаче, то в производной i-й конфигурации k_j-я задача посылает сообщение m_j-й задаче, где k_j-я задача — j-я из i задач, соответствующих k-й задаче конфигурации $CONF_{base}$, m_j-я задача — j-я из i задач, соответствующих m-й задаче конфигурации $CONF_{base}$.Таким образом, в i-й конфигурации в i раз больше задач и в i раз больше сообщений, чем в конфигурации $CONF_{base}$.

Имея данные о характере зависимости времени построения/прогона модели от количества задач (результаты, полученные на наборе 1), а также результаты, полученные на наборе 2, можно сделать вывод о характере зависимости времени построения/прогона модели от количества сообщений.

Исследование зависимости времени построения и прогона модели от количества окон при фиксированном интервале планирования. Для этого исследования был сформирован *набор 3*, содержащий 10 конфигураций МВС, различающихся количеством окон при фиксированном интервале планирования. Конфигурации были получены варьированием параметров «минимальная длительность окна» и «максимальная длительность окна» в САПР «Планировщик ИМА» для рабочей нагрузки и набора ядер, соответствующих реальной системе [19]. Первая конфигурация содержит порядка 4000 окон на интервале планирования, i-я конфигурация — порядка $4000 \cdot i$ окон. Конфигурация $CONF_{base}$ является второй в наборе 3. Так как с помощью варьирования только длительности окна получить конфигурацию с заранее заданным точным числом окон затруднительно, допускается отклонение от числа $4000 \cdot i$ не более, чем на 50.

Исследование зависимости времени построения и прогона модели от количества разделов. Для этого исследования был сформирован *набор 4*, содержащий 10 конфигураций МВС, различающихся количеством разделов. Первая конфигурация набора — конфигурация $CONF_{base}$. i-я конфигурация получена из $CONF_{base}$ разбиением множества задач каждого раздела на i разделов. Для того, чтобы при большом количестве разделов выполнялись ограничения реального времени для конфигурации, необходимо чтобы длительность окон была небольшой. Поэтому длительности окон в конфигурациях этого набора соответствуют длительностям окон в 9-й конфигурации набора 3.

Исследование характера изменения времени построения и прогона модели при согласованном увеличении количества разделов, ядер и окон. Для этого исследования был сформирован *набор 5*, содержащий 10 конфигураций МВС, различающихся количеством разделов, количеством используемых ядер и количеством окон. Первая конфигурация набора — конфигурация $CONF_{base}$. i-я конфигурация получена из $CONF_{base}$ разбиением множества задач каждого раздела на i частей, увеличением количества ядер в i раз и увеличением количества окон в i раз. При этом, если в конфигурации $CONF_{base}$ k-й раздел может быть привязан к m-му ядру, то в производной i-й конфигурации k_j-й раздел может быть привязан к m_j-му ядру, где k_j-й раздел — j-й из i разделов, соответствующих k-му разделу конфигурации $CONF_{base}$, m_j-ое ядро — j-ое из i ядер, соответствующих m-му ядру конфигурации $CONF_{base}$. Для всех разделов производной конфигурации, соответствующих одному разделу исходной конфигурации, наборы ядер, к которым может быть привязан тот или иной раздел, не пересекаются. Длительность окон во всех конфигурациях неизменна, а увеличение количества окон является следствием увеличения количества ядер. Аналогично набору 4, длительности окон в конфигурациях этого набора данных соответствуют длительностям окон в 9-й конфигурации набора 3.

Исследование зависимости времени построения и прогона модели от длительности интервала планирования. Для этого исследования был сформирован *набор 6*, содержащий 10 конфигураций МВС, различающихся длительностью интервала планирования. Первая конфигурация набора — конфигурация $CONF_{base}$ с длительностью интервала планирования 2000 мс,

выбранной для этой системы средствами САПР «Планировщик ИМА». i-я конфигурация получена из $CONF_{base}$ увеличением интервала планирования в i раз.

Увеличение длительности интервала планирования в i раз приводит к увеличению в i раз количества окон в конфигурации, а также количества работ и экземпляров сообщений, т.е. к увеличению в i раз количества моментов времени, в которые в МВС происходят события, а в модели МВС — дискретные переходы. В отличие от набора 6, в конфигурациях набора 3 количество работ и экземпляров сообщений остается неизменным.

Исследование характера изменения времени построения и прогона модели при согласованном увеличении всех характеристик конфигурации. Для этого исследования был сформирован *набор 7*, содержащий 10 конфигураций МВС таких, что первой конфигурацией набора является $CONF_{base}$, а i-я конфигурация получена из $CONF_{base}$ i-кратным дублированием. Таким образом, i-я конфигурация содержит в i раз больше задач, сообщений, разделов, ядер и окон, чем первая.

Исследование зависимости времени построения и прогона модели от временных характеристик функциональных задач: смещений, задающих директивные интервалы работ, и логического значения условия выполнения ограничений реального времени. Для этого исследования были сформированы наборы 8—10, аналогичные набору 7 и отличающиеся от него лишь способами выбора смещений относительно начала периода, задающих границы директивных интервалов работ. В конфигурациях набора 7 смещение, задающее левые границы директивных интервалов работ задачи, равно 0, смещение, задающее правые границы директивных интервалов работ задачи, равно периоду этой задачи. Это особенность реальной МВС, имеющей конфигурацию $CONF_{base}$. В то же время, согласно разделу 2.8, сложность прогона модели зависит от количества моментов времени, соответствующих дискретным переходам в модели. Использование смещений, отличных от 0 и периода соответственно, увеличивает количество таких моментов времени. Кроме того, опоздание некоторых работ приводит к тому, что ни одна из других работ, зависящих по данным от опоздавших, не ставится на выполнение, что ведет к уменьшению времени прогона модели.

В качестве базовых конфигураций при формировании конфигураций наборов 8—10 были выбраны конфигурации набора 7, т.к., по сравнению с остальными наборами, конфигурации этого набора различаются между собой наибольшим количеством характеристик.

В конфигурациях *набора 8* смещения, задающие правые границы директивных интервалов работ, не совпадают с периодами задач, при этом все работы успевают завершиться в срок. i-я конфигурация набора 8 получена из i-й конфигурации набора 7 следующим образом: для каждой задачи с помощью исследуемой модели МВС получено максимальное по всем работам время завершения относительно начала периода. Полученное значение, увеличенное на 1, устанавливается в качестве смещения, задающего правые границы директивных интервалов работ задачи.

В конфигурациях *набора 9* смещения, задающие левые границы директивных интервалов работ, не равны 0, а смещения, задающие правые границы директивных интервалов работ, не совпадают с периодами задач, т.е. директивный интервал каждой работы лежит строго внутри интервала, заданного периодом задачи и номером работы. i-я конфигурация набора 9 получена из i-й конфигурации набора 7 следующим образом: для каждой задачи смещение, задающее левые

границы директивных интервалов работ, равно 0.5% WCET этой задачи, а смещение, задающее правые границы директивных интервалов работ, равно периоду, уменьшенному на 0.5% WCET. С помощью исследуемой модели МВС и анализатора ВД проверено, что все конфигурации удовлетворяют ограничениям реального времени.

Примечание: при изменении рассматриваемых смещений, более чем на 0.5% WCET, некоторые работы не успевают штатно завершиться в рамках директивных интервалов. Так при изменении смещений в конфигурации $CONF_{base}$ на 1% WCET, ограничения реального времени не выполняются для двух задач конфигурации.

В конфигурациях *набора 10* смещения, задающие правые границы директивных интервалов работ таковы, что ни для одной работы ни одной задачи не выполняются ограничения реального времени. i-я конфигурация набора 10 получена из i-й конфигурации набора 7 следующим образом: для каждой задачи с помощью исследуемой модели МВС получено минимальное по всем работам время завершения относительно начала периода. Полученное значение, уменьшенное на 1, устанавливается в качестве смещения, задающего правые границы директивных интервалов работ задачи.

5.2. Анализ результатов экспериментов

В данном разделе приведены и проанализированы результаты проведенных экспериментов, представленные в виде графиков. Обозначение *x<i>* на графике соответствует i-й конфигурации набора. Результаты экспериментов в табличном виде приведены в приложении Г.

5.2.1. Зависимость времени построения экземпляра модели от характеристик конфигурации системы

Согласно разделу 2.8, сложность построения экземпляра модели имеет порядок $O(N + \sum_{i=1}^{M} N_i^w + K + H)$, где N — количество ядер, $\sum_{i=1}^{M} N_i^w$ — количество окон, K — количество задач, H — количество сообщений.

Результаты экспериментов для наборов 1 и 2 приведены на рисунке 5.1. Также на рисунке 5.1 приведена разность времени построения экземпляра модели для соответствующих конфигураций наборов 2 и 1, позволяющая оценить характер зависимости этого времени только от количества сообщений. Для каждой из трех полученных зависимостей методом наименьших квадратов были построена линейная функция регрессии, для которой коэффициент детерминации равен соответственно 0.99, 0.99 и 0.94. Значения F-статистики равны 6517, 5089 и 139 соответственно при табличном значении 5.32 (методика расчета коэффициента детерминации и F-статистики

приведена в разделе 5.1) Таким образом, экспериментальные данные подтверждают порядок теоретических оценок сложности построения экземпляра модели в зависимости от количества задач и сообщений, т.е. эта сложность может быть выражена построенной линейной функцией.

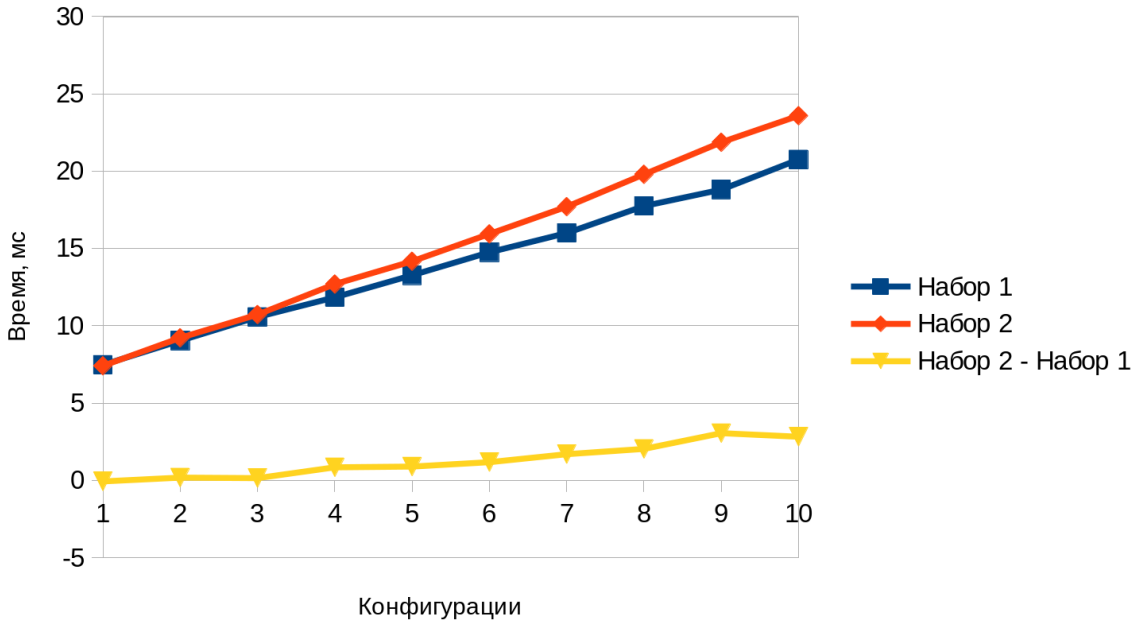

Рисунок 5.1 — Зависимость времени построения экземпляра модели от количества задач (набор 1) и суммарного количества задач и сообщений (набор 2) в конфигурации

Результаты экспериментов для набора 3 приведены на рисунке 5.2. Также на этом рисунке приведены результаты экспериментов для набора 6, поскольку с увеличением в n раз длительности интервала планирования увеличивается в n раз и количество окон, а остальные характеристики конфигурации, входящие в формулу оценки сложности построения экземпляра модели, остаются неизменными (в отличие от оценки сложности прогона). Таким образом, характер зависимостей, полученных на наборах 3 и 6, должен совпадать. Так как конфигурации наборов 3 и 6, имеющие один номер, различаются по количеству окон (по количеству окон i-й конфигурации набора 6 соответствует $2 \cdot i$-я конфигурация набора 3), то на горизонтальной оси графика приведено количество окон, а конкретным конфигурациям соответствуют точки на графике.

По экспериментальным данным, полученным на каждом из наборов отдельно, а также по совокупности данных, методом наименьших квадратов были построены линейные функции регрессии, для которых коэффициенты детерминации равны 0.99 (одинаковы для всех трех функций), а значения F-статистики равны 3636 (набор 3), 876 (набор 6) и 1276 (совокупность данных двух наборов) при табличных значениях 5.32 (для данных по наборам отдельно) и 4.41 (для совокупности данных). Таким образом, экспериментальные данные подтверждают порядок теоретических оценок сложности построения экземпляра модели в зависимости от количества окон.

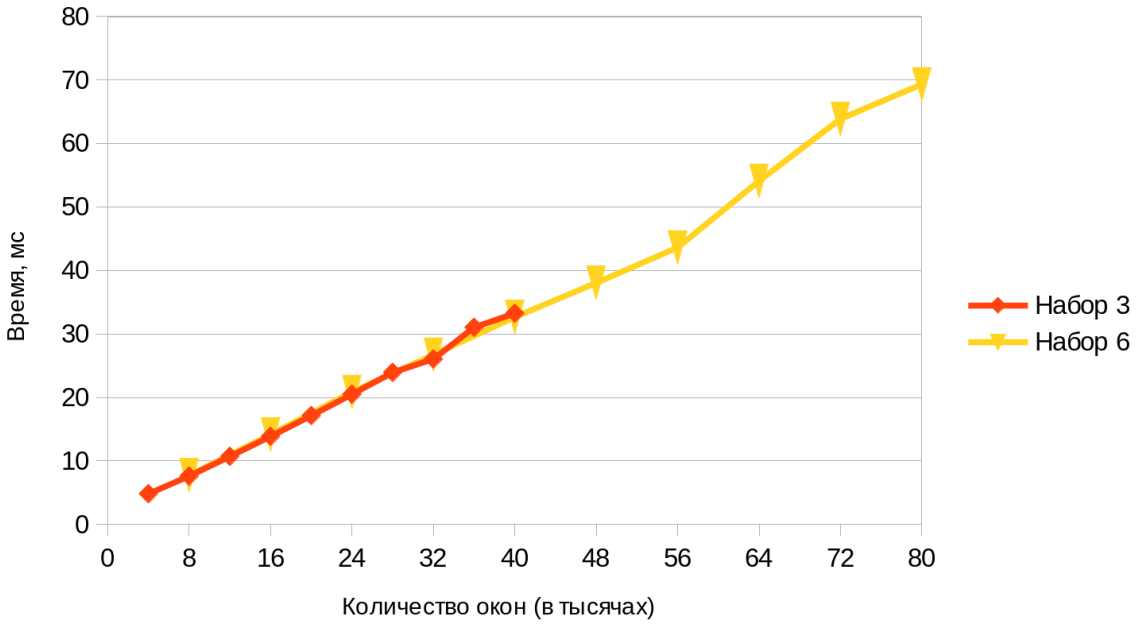

Рисунок 5.2 — Зависимость времени построения экземпляра модели от количества окон (набор 3) и длительности интервала планирования (набор 6)

Результаты экспериментов для набора 4 приведены на рисунке 5.3. На основе полученных экспериментальных данных методом наименьших квадратов была построена линейная функция регрессии, для которой коэффициент детерминации равен 0.79, а значение F-статистики равно 32 при табличном значении 5.32. Т.к. вычисленное значение F-статистики больше табличного значения, то полученная регрессия считается значимой, несмотря на меньшее, чем в случае других наборов данных, значение коэффициента детерминации. Согласно разделу 2.8 сложность построения экземпляра модели имеет порядок $O(N + M + \sum_{i=1}^{M} N_i^w + K + H) = O(N + \sum_{i=1}^{M} N_i^w + K + H)$, поскольку $M \leqslant K$ (M — количество разделов для конфигурации). В серии экспериментов на конфигурациях набора 4 все характеристики конфигураций, кроме количества разделов, фиксированы. Поэтому для конфигураций набора 4 время построения экземпляра модели в худшем случае должно расти линейно с увеличением номера конфигурации. Таким образом, экспериментальные данные подтверждают порядок теоретических оценок сложности построения экземпляра модели в зависимости от количества окон.

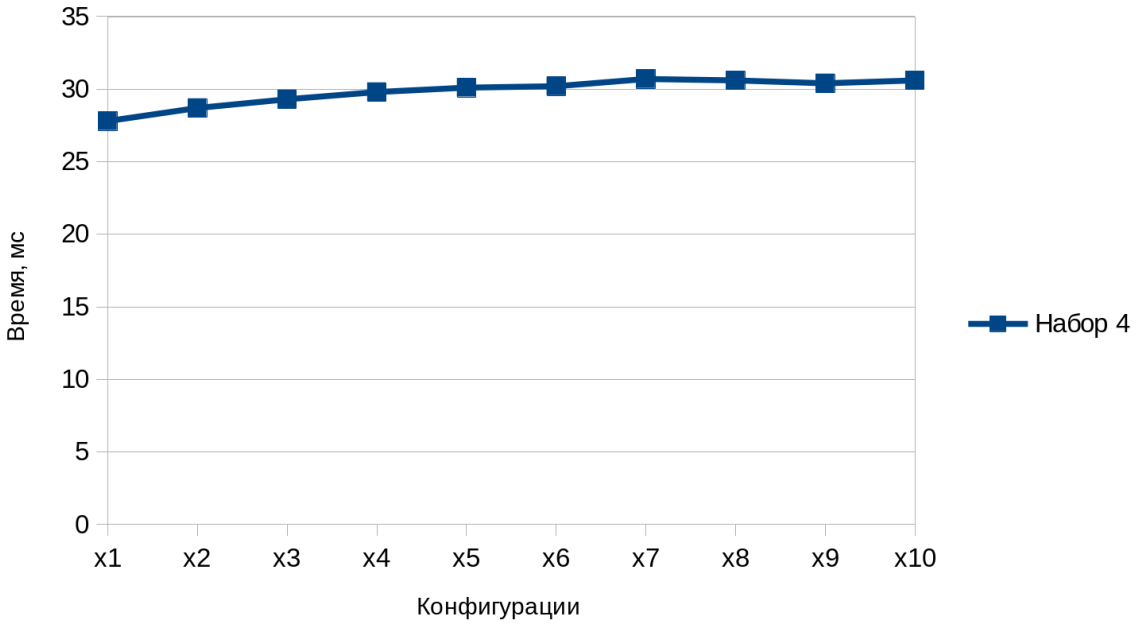

Рисунок 5.3 — Зависимость времени построения экземпляра модели от количества разделов в конфигурации

Результаты экспериментов для набора 5 приведены на рисунке 5.4. На основе полученных экспериментальных данных методом наименьших квадратов была построена линейная функция регрессии, для которой коэффициент детерминации равен 0.99, а значение F-статистики равно 10590 при табличном значении 5.32. Таким образом, экспериментальные данные подтверждают порядок теоретических оценок сложности построения экземпляра модели в зависимости от кратности согласованного увеличения количества разделов, окон и ядер в конфигурации.

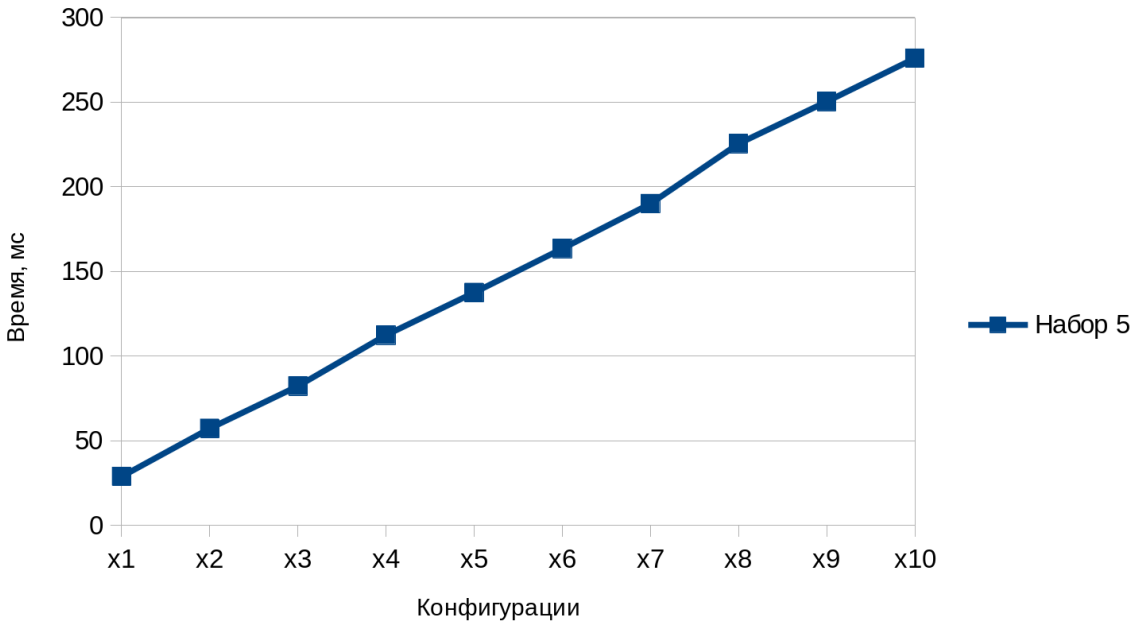

Рисунок 5.4 — Зависимость времени построения экземпляра модели от кратности согласованного увеличения количества разделов, окон и ядер в конфигурации

Результаты экспериментов для наборов 7—10 приведены на рисунке 5.5. i-кратное дублирование системы приводит к увеличению числа задач, сообщений, разделов, ядер и окон в i раз. На основе полученных экспериментальных данных для набора 7 методом наименьших квадратов была построена линейная функция регрессии, для которой коэффициент детерминации равен 0.99, а значение F-статистики равно 27139 при табличном значении 5.32. Таким образом, экспериментальные данные подтверждают порядок теоретических оценок сложности построения экземпляра модели в зависимости от суммарного количества задач, сообщений, разделов, ядер и окон.

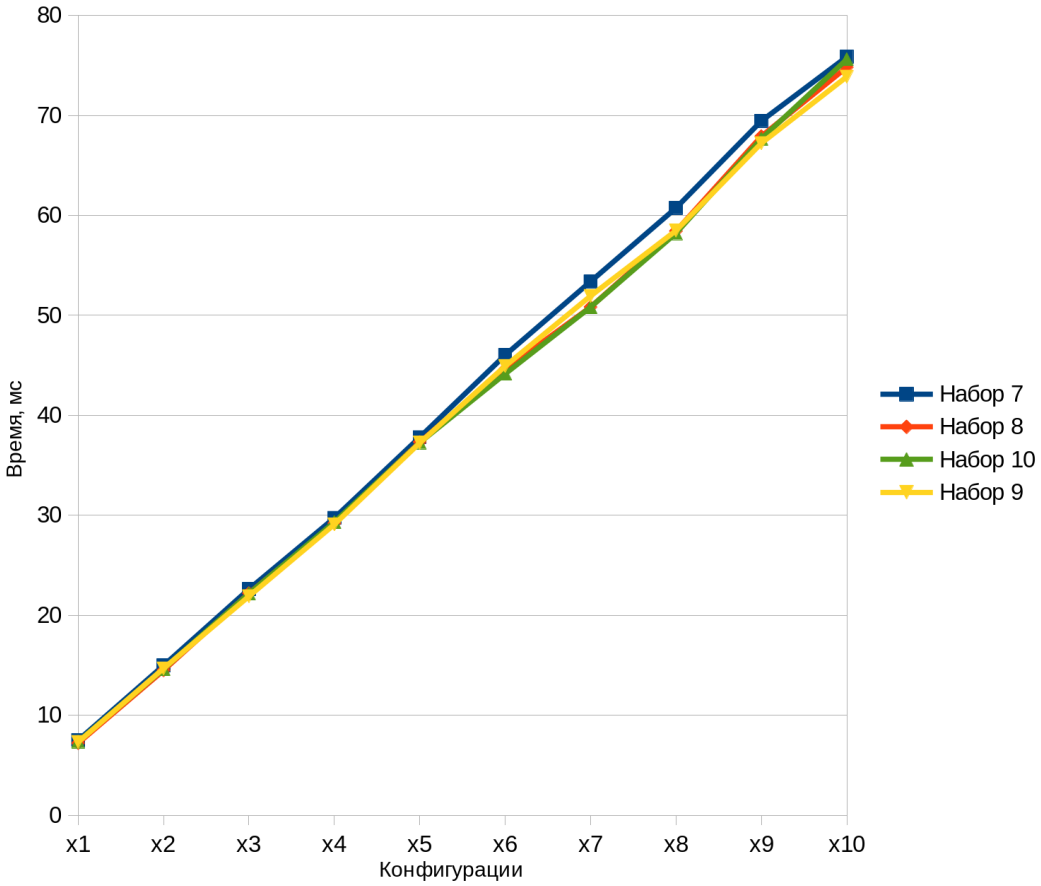

Рисунок 5.5 — Зависимость времени построения экземпляра модели от кратности дублирования системы для различных временных характеристик задач

Значения времени построения экземпляра модели для конфигураций одинаковой размерности наборов 7—10 практически не различаются, что свидетельствует об отсутствии зависимости этого времени от временных характеристик задач. Этот вывод совпадает с теоретическими рассуждениями.

5.2.2. Зависимость времени прогона экземпляра модели от характеристик конфигурации системы

Согласно разделу 2.8 сложность прогона модели имеет порядок $O(L \cdot (K^2 + H^2))$, где L — длительность интервала планирования, K — количество задач, H — количество сообщений.

В этой формуле значением L оценено сверху количество моментов времени на интервале планирования, соответствующих дискретным переходам в модели. Количество таких моментов времени зависит от количества окон. Кроме того, на него могут влиять соотношения временных характеристик функциональных задач. Так, например, для конфигурации МВС, содержащей одну задачу с ненулевым смещением, задающим левые границы директивных интервалов работ, количество таких моментов времени больше, чем для аналогичной конфигурации, содержащей задачу, имеющую нулевое смещение.

Результаты экспериментов для наборов 1 и 2 приведены на рисунке 5.6. Также на рисунке 5.6 приведена разность времени прогона модели для соответствующих конфигураций наборов 1 и 2, позволяющая оценить характер зависимости этого времени только от количества сообщений.

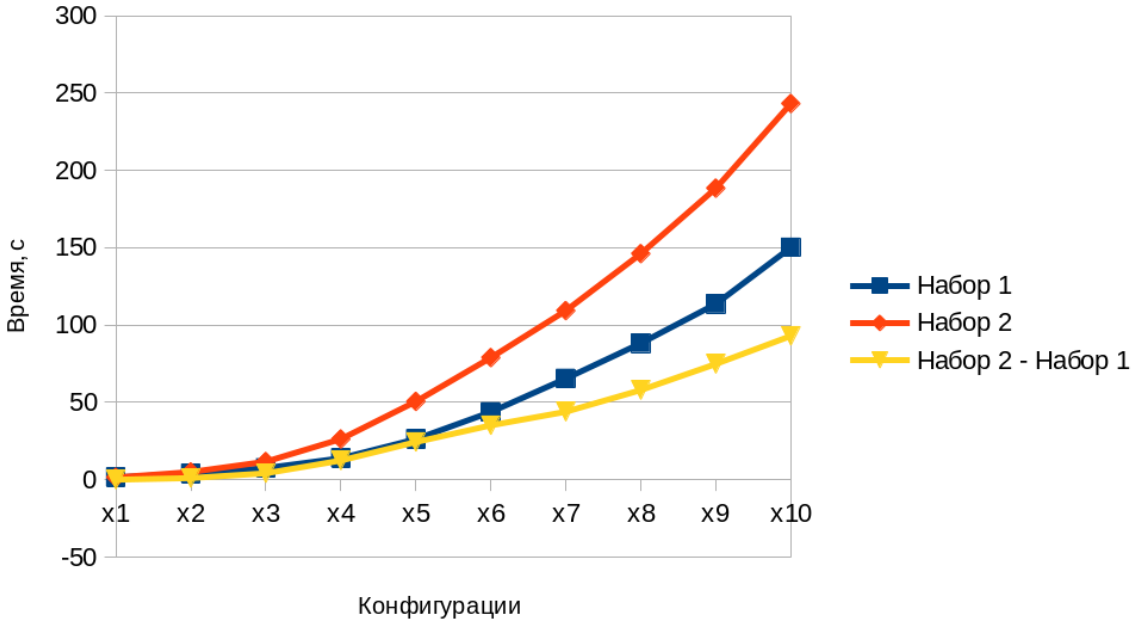

Рисунок 5.6 — Зависимость времени прогона модели от количества задач (набор 1) и суммарного количества задач и сообщений (набор 2) в конфигурации

Увеличение числа задач в i раз при фиксированных остальных характеристиках конфигурации посредством разбиения каждой задачи конфигурации $CONF_{base}$ на i задач приводит к увеличению количества моментов времени, соответствующих дискретным переходам в модели, не более чем в i раз. Обоснуем это. Дискретные переходы в моделях задачи соответствуют следующим моментам времени:

– Границы директивных интервалов работ. Все добавленные задачи имеют период, равный периоду соответствующей задачи в конфигурации $CONF_{base}$, а смещения, задающие левые и правые границы директивных интервалов работ задачи равны 0 и периоду задачи соответственно. Следовательно, границы директивных интервалов работ добавленных задач совпадают с границами директивных интервалов работ соответствующих задач конфигурации $CONF_{base}$ и новых границ директивных интервалов не добавляется.
– Моменты получения и отправки сообщений. Все добавленные задачи не имеют зависимостей по данным, поэтому количество таких моментов не увеличивается.

– Моменты появления готовых работ. Все добавленные задачи не имеют зависимостей по данным и имеют период, равный периоду соответствующей задачи в конфигурации $CONF_{base}$. Поэтому все работы добавленных задач становятся готовыми в начале периода и, следовательно, новых моментов времени, соответствующих появлению готовой работы, не добавляется.
– Моменты завершения работ. При разбиении каждой задачи конфигурации $CONF_{base}$ на i задач количество таких моментов времени увеличивается в i раз. В худшем случае все эти моменты являются новыми, т.е. не совпадают с моментами других событий.
– Моменты постановки работ на выполнение. Работа может быть поставлена на выполнение либо в момент, когда она становится готовой (например, если используется планировщик с фиксированными приоритетами и вытеснением, и работа имеет максимальный приоритет среди готовых работ), либо в момент, когда завершается другая работа, либо в момент открытия окна раздела. Моменты завершения работ и появления готовых работ уже были учтены ранее, а количество моментов открытия окон при добавлении новых задач не увеличивается.
– Моменты вытеснения работ. Работа может быть вытеснена либо в момент появления новой готовой работы, либо в момент закрытия текущего окна. Моменты появления работ были учтены ранее, а количество моментов закрытия окон при добавлении новых задач не увеличивается.

Новых моментов дискретных переходов, отличных от рассмотренных выше моментов дискретных переходов в моделях задач, в моделях других компонентов МВС не добавляется, т.к. с временными характеристиками задач (именно эти характеристики определяют рассматриваемые моменты времени) работают только модели задач.

Таким образом, количество моментов времени, соответствующих дискретным переходам в модели, в худшем случае увеличивается в i раз при разбиении каждой задачи конфигурации $CONF_{base}$ на i задач.

Аналогичным образом обосновывается, что увеличение числа задач и числа сообщений в i раз при формировании i-й конфигурации набора 2 приводит к увеличению количества моментов времени, соответствующих дискретным переходам в модели, не более чем в $2 \cdot i$ раз.

Таким образом, согласно приведенной выше формуле оценки сложности прогона модели, для наборов 1 и 2 ожидается в худшем случае кубическая зависимость времени прогона модели от числа задач (для набора 1) и от суммарного числа задач и сообщений (для набора 2).

На основе полученных экспериментально результатов для каждого из наборов, а также для разности времени прогона модели для соответствующих конфигураций набора 2 и набора 1, методом наименьших квадратов была построена кубическая функция регрессии, для которой коэффициент детерминации равен 0.99 (одинаков для всех трех функций). Значения F-статистики равны 2830, 4153 и 862 соответственно при табличном значении 4.76. Таким образом, экспериментальные данные подтверждают порядок теоретических оценок сложности прогона экземпляра модели в зависимости от количества задач и сообщений.

Результаты экспериментов для набора 3 приведены на рисунке 5.7. Согласно разделу 2.8 и рассуждениям в начале текущего раздела (см. стр. 107), время прогона модели линейно зависит

от количества моментов времени на интервале планирования, соответствующих дискретным переходам в модели. Увеличение числа окон в i раз при фиксированных остальных характеристиках конфигурации посредством сокращения размера окна в i раз приводит к увеличению количества таких моментов не более чем в $2 \cdot i$ раз, т.к. с разделами работает только модель планировщика ядра, а в ней дискретные переходы соответствуют только моментам открытия и закрытия окон. В худшем случае каждому новому окну соответствует два новых момента времени, в каждый из которых происходит только открытие/закрытие этого окна (событий, связанных с другими окнами, директивными интервалами работ и периодами задач, в худшем случае в этот момент нет). Поэтому для конфигураций набора 3 время прогона модели в худшем случае должно расти линейно с увеличением номера конфигурации.

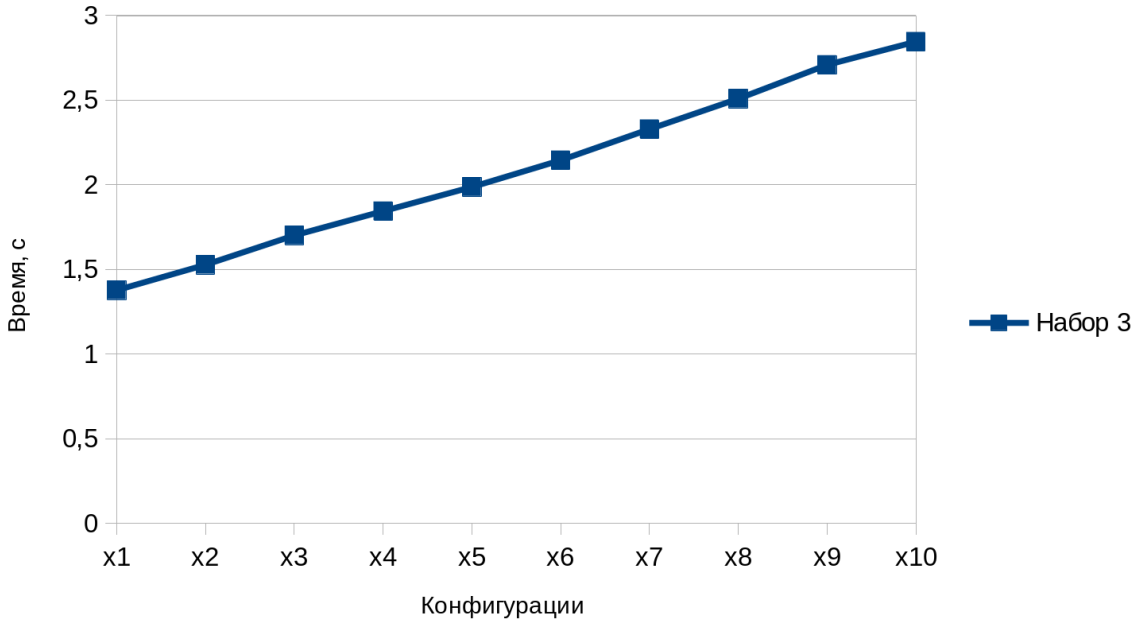

Рисунок 5.7 — Зависимость времени прогона экземпляра модели от количества окон в конфигурации

На основе полученных экспериментальных данных методом наименьших квадратов была построена линейная функция регрессии с коэффициентом детерминации 0.99 и значением F-статистики 4207 при табличном значении 5.32. Таким образом, экспериментальные данные подтверждают порядок теоретических оценок сложности прогона экземпляра модели в зависимости от количества окон.

Результаты экспериментов для набора 4 приведены на рисунке 5.8. Согласно разделу 2.8 сложность прогона модели имеет порядок $O(L \cdot (M + K^2 + H^2)) = O(L \cdot (K^2 + H^2))$, поскольку $M \leqslant K$. В серии экспериментов на конфигурациях набора 4 все характеристики конфигураций, кроме количества разделов, фиксированы. В т.ч. фиксированы количество задач (K), количество сообщений (H), количество моментов времени, соответствующих дискретным переходам в модели. Поэтому для конфигураций набора 4 время прогона модели в худшем случае должно расти линейно с увеличением номера конфигурации.

На основе полученных экспериментальных данных методом наименьших квадратов была построена линейная функция регрессии, для которой коэффициент детерминации равен 0.99,

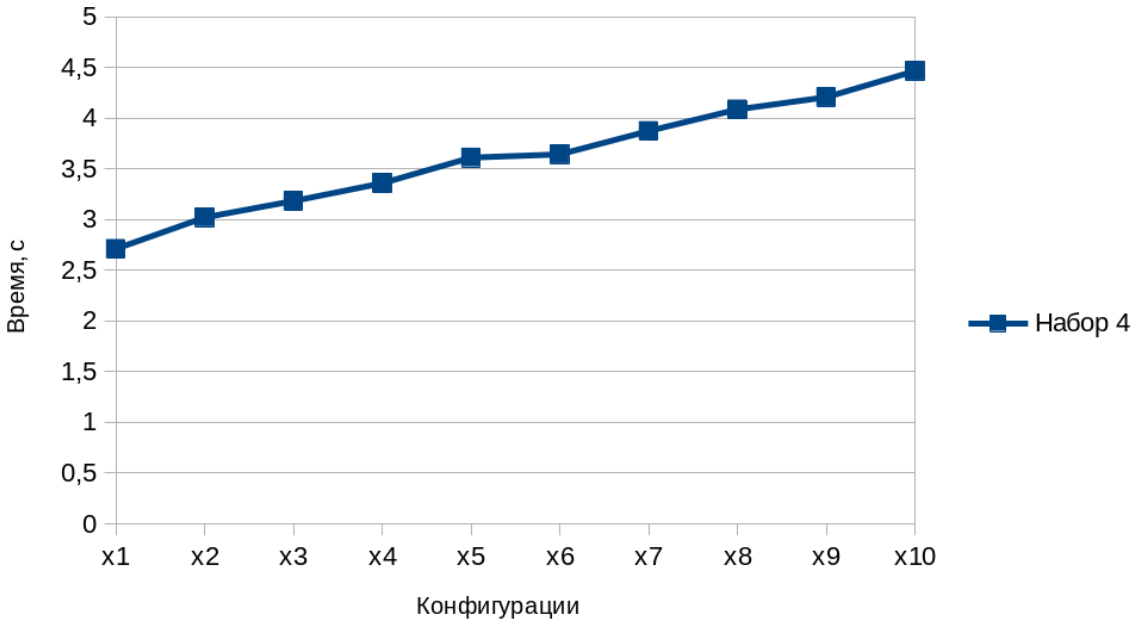

Рисунок 5.8 — Зависимость времени прогона экземпляра модели от количества разделов в конфигурации

а значение значением F-статистики равно 887 при табличном значении 5.32. Таким образом, экспериментальные данные подтверждают порядок теоретических оценок сложности прогона экземпляра модели в зависимости от количества разделов.

Результаты экспериментов для набора 5 приведены на рисунке 5.9. При получении i-й конфигурации на основе конфигурации $CONF_{base}$ в i раз увеличиваются количество разделов, количество ядер (посредством увеличения в i раз количества модулей) и количество окон. Поскольку при разбиении набора задач одного раздела в базовой конфигурации на несколько разделов в производной конфигурации в разные разделы попадают задачи с различными временными характеристиками, то в производной конфигурации моменты открытия и закрытия соответствующих окон для ядер разных модулей не совпадают. В худшем случае одному моменту открытия некоторого окна в конфигурации $CONF_{base}$ соответствует i разных моментов открытия соответствующих этому окну окон в производной конфигурации (аналогично для моментов закрытия окон). Таким образом, при получении i-й конфигурации на основе конфигурации $CONF_{base}$ в не более чем i раз увеличивается количество моментов времени, соответствующих дискретным переходам в модели (в формуле для оценки сложности прогона модели эта величина ограничена сверху значением L), и в i раз увеличивается количество разделов при фиксированном количестве задач и сообщений. Поэтому, согласно формуле $O(L \cdot (M + K^2 + H^2))$, для конфигураций набора 5 время прогона модели в худшем случае должно расти квадратично с увеличением номера конфигурации.

На основе полученных экспериментальных данных методом наименьших квадратов была построена квадратичная функция регрессии, для которой коэффициент детерминации равен 0.99, а значение F-статистики равно 3346 при табличном значении 4.74. Таким образом, экспериментальные данные подтверждают порядок теоретических оценок сложности прогона экземпляра модели в зависимости от кратности согласованного увеличения количества разделов, окон и ядер

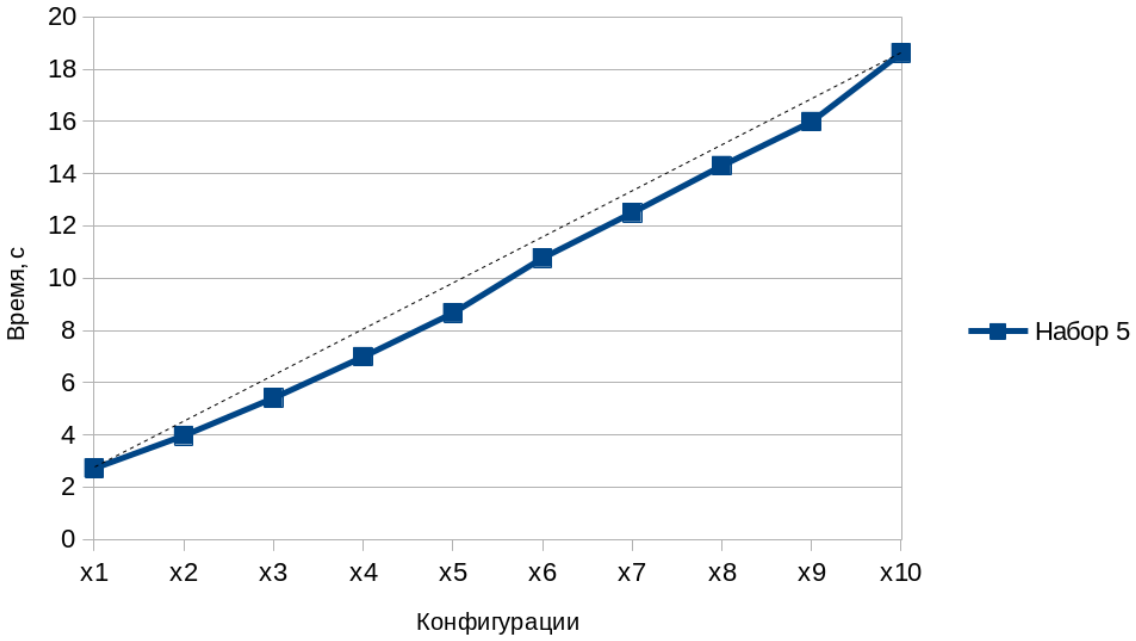

Рисунок 5.9 — Зависимость времени прогона модели от кратности согласованного увеличения количества разделов, окон и ядер в конфигурации

в конфигурации. Согласно приведенному на рисунке 5.9 графику, построенная парабола имеет небольшую степень крутизны (для того, чтобы показать нелинейность графика, конечные точки соединены прямой), что говорит о том, что при построении i-й конфигурации на основе конфигурации $CONF_{base}$, количество моментов времени, соответствующих дискретным переходам в модели (в формуле для оценки порядка сложности прогона экземпляра модели это количество оценено сверху длительностью интервала планирования L), увеличивается менее, чем в i раз. Это может быть связано с тем, что из-за узости интервала, которому принадлежит длительность окна (от 332 до 341 мс), моменты открытия и закрытия многих окон на разных ядрах совпадают.

Результаты экспериментов для набора 6 приведены на рисунке 5.10. Согласно приведенной выше оценке, время прогона модели линейно зависит от длительности интервала планирования: с увеличением длительности интервала планирования в i раз при фиксированных временных характеристиках задач и сообщений количество моментов времени, соответствующих дискретным переходам в модели, возрастет в i раз.

На основе полученных экспериментальных данных методом наименьших квадратов была построена линейная функция регрессии, для которой коэффициент детерминации равен 0.99, а значением F-статистики равно 139312 при табличном значении 5.32. Таким образом, экспериментальные данные подтверждают порядок теоретических оценок сложности прогона экземпляра модели в зависимости от длительности интервала планирования.

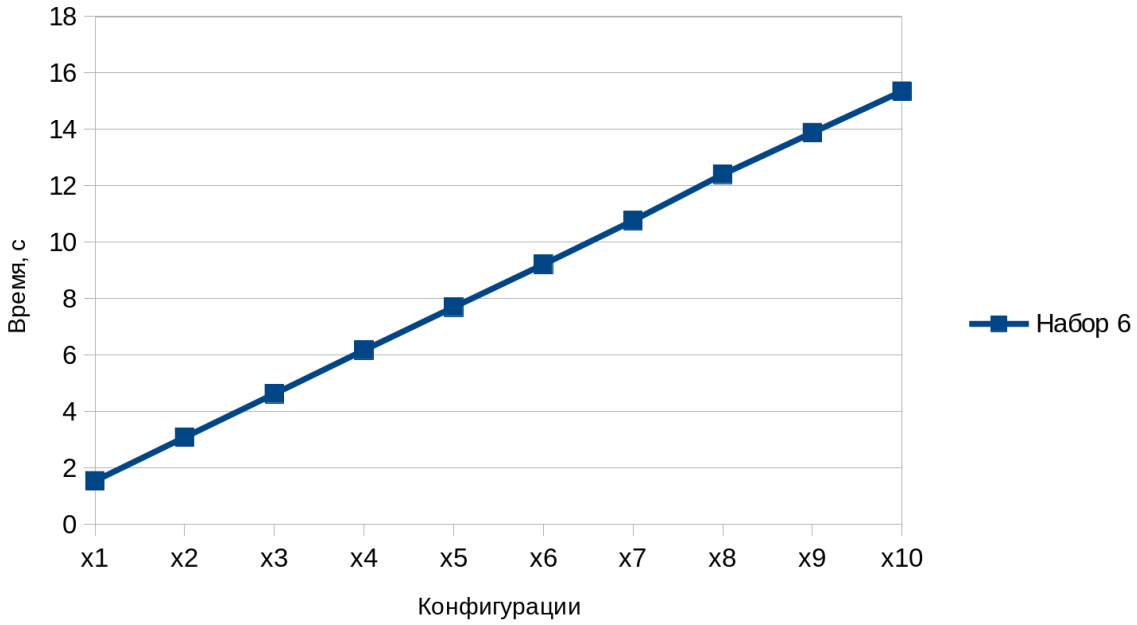

Рисунок 5.10 — Зависимость времени прогона экземпляра модели от длительности интервала планирования

Результаты экспериментов для набора 7 приведены на рисунке 5.11. i-кратное дублирование системы приводит к увеличению числа задач, сообщений, разделов, ядер и окон в i раз. При этом увеличение этих характеристик не приводит к увеличению количества моментов времени на интервале планирования, соответствующих дискретным переходам в модели, так как все «копии» системы функционируют синхронно. Поэтому, согласно формуле $O(L \cdot (K^2 + H^2))$, для конфигураций набора 5 время прогона модели в худшем случае должно расти квадратично с увеличением номера конфигурации.

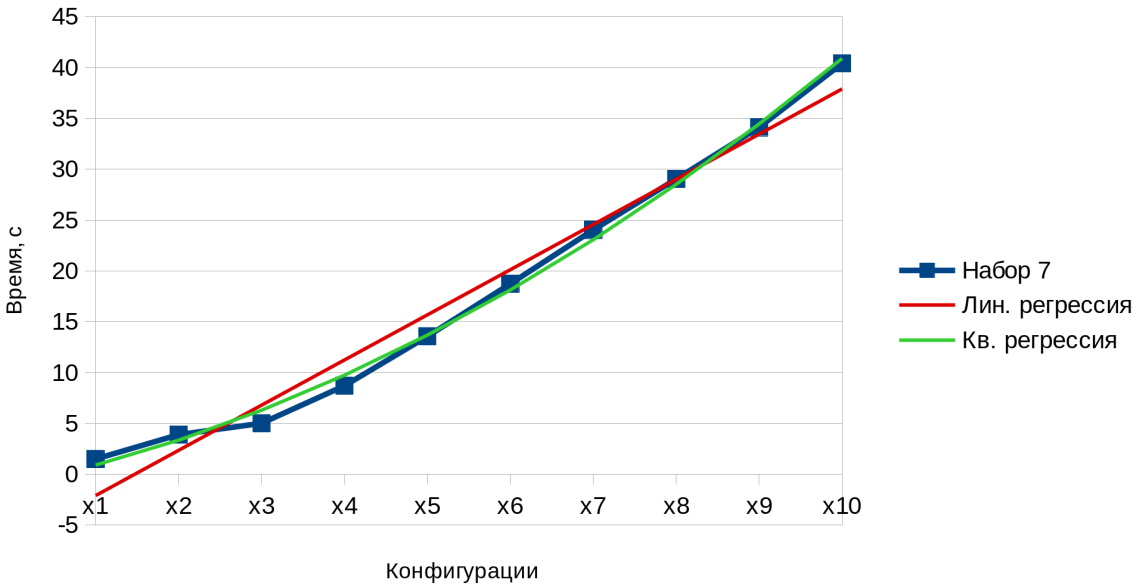

Рисунок 5.11 — Зависимость времени прогона экземпляра модели от кратности дублирования системы

На основе полученных экспериментальных данных методом наименьших квадратов была построена линейная функция регрессии, для которой коэффициентом детерминации равен 0.99, а значение F-статистики равно 15785 при табличном значении 5.32, и квадратичная функция регрессии, для которой коэффициент детерминации равен 0.99, а значение F-статистики равно 1093 при табличном значении 4.74. На рисунке 5.11 приведены обе линии регрессии. Высокий коэффициент детерминации для линейной функции регрессии и низкая степень крутизны параболы объясняются пессимистичностью теоретической оценки сложности прогона модели (при ее выводе рассматривался худший с точки зрения производительности набор характеристик конфигурации), что свойственно для верхних оценок.

Результаты экспериментов для наборов 8—10 в сравнении с результатами для набора 7 приведены на рисунке 5.12.

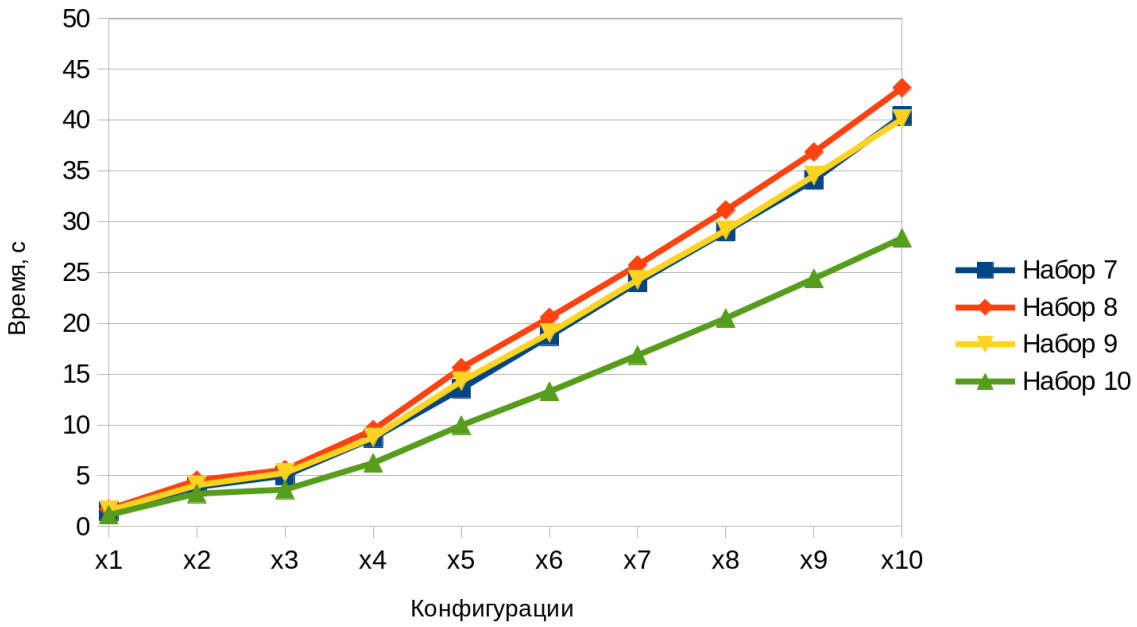

Рисунок 5.12 — Зависимость времени прогона экземпляра модели от кратности дублирования системы для различных значений границ директивных интервалов работ

Указание для задач смещений, задающих правые границы директивных интервалов работ, отличных от периодов (набор 8), приводит к увеличению количества моментов времени на интервале планирования, соответствующих дискретным переходам в модели, поэтому время прогона экземпляра модели для конфигураций наборов 8 превышает время прогона для соответствующих конфигураций набора 7.

Сужение директивных интервалов работ (набор 9) также приводит к увеличению количества указанных моментов времени, однако максимальная возможная величина смещения границы каждого директивного интервала составляет всего 0.5% WCET. Поскольку значения всех временных характеристик задач округляются до целого числа микросекунд, то для некоторых задач 0.5% WCET равно 0 и реального сужения директивных интервалов не происходит. Для ряда задач величина 0.5% WCET имеет одно и то же значение, в результате директивные интервалы работ таких задач совпадают. Поэтому выполненное сужение директивных интервалов работ привело

к увеличению количества моментов времени на интервале планирования, соответствующих дискретным переходам в модели, всего на 6% по сравнению с конфигурациями набора 7. В результате длительности прогона моделей для соответствующих конфигураций наборов 7 и 9 практически совпадают.

Нарушение ограничений реального времени (набор 10) приводит к тому, что при завершении работы в связи с достижением правой границы ее директивного интервала не происходит отправка сообщений. Следовательно, работы задач-получателей никогда не становятся готовыми и, таким образом, значительная часть компонентов модели «простаивает». Поэтому время прогона экземпляра модели для конфигураций набора 10 меньше времени прогона для соответствующих конфигураций наборов 7—9.

5.2.3. Сравнение по производительности разработанной параметризованной модели МВС с моделью, встроенной в САПР «Планировщик ИМА»

Результаты сравнения разработанной параметризованной модели МВС со встроенной в САПР «Планировщик ИМА» моделью по времени проверки выполнения ограничений реального времени на конфигурациях набора 7 приведены на рисунке 5.13. В отличие от экспериментов, результаты которых приведены в разделах 5.2.2 и 5.2.1, в данной серии экспериментов для разработанной модели замерялась полная длительность проверки выполнения ограничений реального времени. Эта проверка включает генерацию файла с описанием конфигурации в САПР, обращение из САПР к исполняемой параметризованной модели МВС для построения и прогона экземпляра модели, ожидание завершения прогона и разбор в САПР файла, содержащего ВД.

Результаты экспериментов показали, что на конфигурациях небольшой размерности (1-я конфигурация набора 7, соответствующая реальной МВС [19], и 2-я конфигурация набора 7) встроенная в САПР «Планировщик ИМА» модель незначительно превосходит разработанную параметризованную модель по производительности. Однако, с ростом размерности конфигураций разработанная модель начинает работать заметно быстрее, чем модель из САПР «Планировщик ИМА». Так на 10-й конфигурации набора 7 разработанная модель превосходит по скорости работы встроенную в САПР «Планировщик ИМА» модель в три раза, и позволяет проверить выполнение ограничений реального времени на 85 секунд быстрее. Также эксперименты показали, что время работы встроенной в САПР «Планировщик ИМА» модели с ростом размерности конфигураций растет значительно быстрее, чем время работы разработанной параметризованной модели МВС. ВД, построенные с помощью обеих моделей, совпадают.

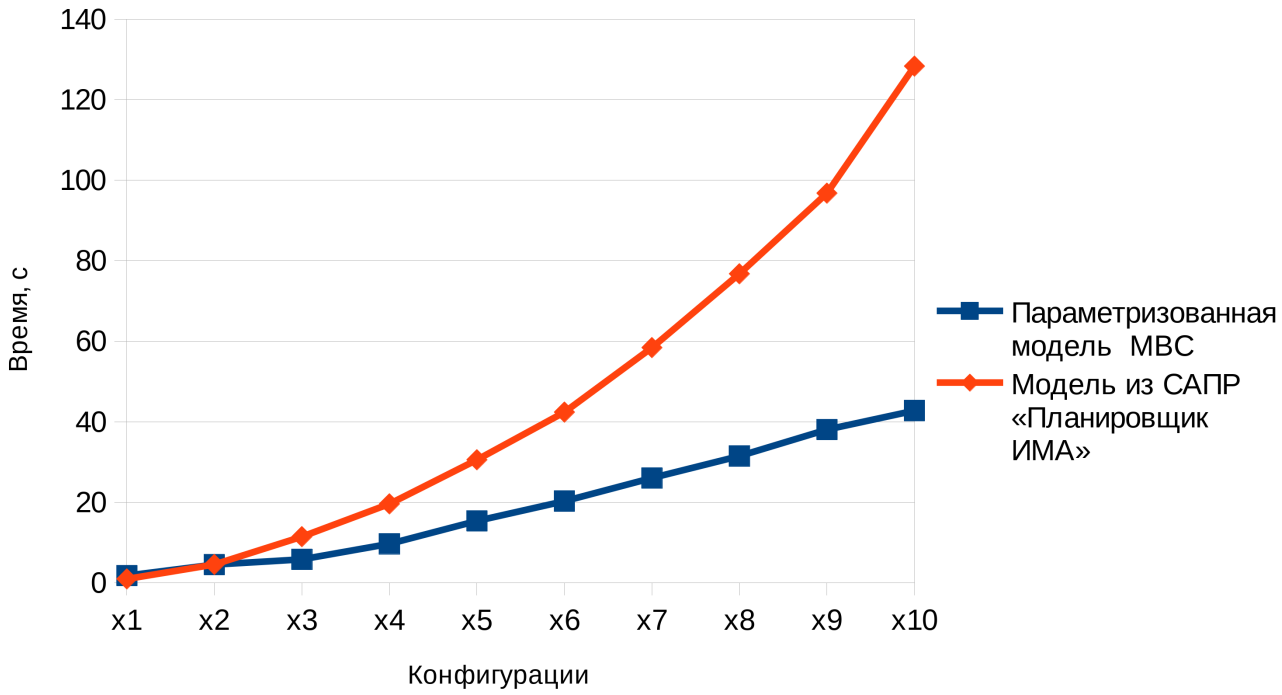

Рисунок 5.13 — Сравнение времени работы параметризованной модели МВС и встроенной в САПР «Планировщик ИМА» модели

5.3. Выводы

Результаты экспериментов показали, что фактический порядок зависимости длительностей построения/прогона экземпляров моделей от характеристик конфигураций МВС соответствуют порядку теоретических оценок сложности построения/прогона экземпляров моделей, приведенных в разделе 2.8.

Для конфигурации размерности, соответствующей реальной МВС [19], время проверки ограничений реального времени составило менее двух секунд (на используемой для экспериментов машине), что позволяет за сутки обработать несколько десятков тысяч конфигураций. Следовательно, для конфигураций такой размерности разработанное средство может быть использовано в цикле работы поисковых алгоритмов, перебирающих большое число потенциальных конфигураций (например, эволюционных алгоритмов).

Сравнение разработанного средства с моделью, встроенной в САПР «Планировщик ИМА», показало, что на конфигурациях небольшой размерности встроенная модель незначительно превосходит по производительности разработанное средство. Однако с ростом размерности конфигураций разработанное средство начинает превосходить встроенную модель, поскольку время работы встроенной в САПР «Планировщик ИМА» модели с ростом размерности конфигураций растет значительно быстрее, чем время работы разработанной параметризованной модели МВС: чем больше размерность, тем больше преимущество разработанной модели.

Заключение

Основные результаты диссертационной работы заключаются в следующем:

1. Построена обобщенная модель функционирования МВС, абстрагированная от структуры МВС и используемых в МВС алгоритмов планирования. Модель базируется на аппарате сетей временных автоматов с остановкой таймеров, расширенном в работе для абстрагирования от систем переходов автоматов. Доказана корректность обобщенной модели функционирования МВС.
2. Разработан метод проверки выполнения ограничений реального времени для заданной конфигурации МВС. Метод конкретизирует обобщенную модель функционирования МВС для заданной конфигурации и использует полученную модель при построении временной диаграммы функционирования МВС.
3. На основе предложенного метода разработана инструментальная система проверки выполнения ограничений реального времени для конфигураций МВС. Экспериментальное исследование подтвердило применимость разработанного метода для анализа конфигураций МВС реальной размерности.

Можно выделить следующие направления дальнейших исследований, а также применения полученных в работе результатов:

- Использование и развитие разработанных методов и средств для анализа конфигураций вычислительных систем других классов, например, сочетающих элементы интегрированной модульной архитектуры с элементами федеративной архитектуры.
- Обобщение и исследование применимости предложенного метода обоснования корректности параметризованных автоматов-моделей компонентов МВС (в т.ч. подхода к сокращению диапазонов перебираемых при верификации значений параметров) к верификации параметризованных автоматов-моделей компонентов компьютерных сетей, например — абонентов протоколов с внутренним состоянием.
- Разработка и исследование применимости новых уровней абстракции, аналогичных обобщенным сетям временных автоматов с остановкой таймеров, для других математических аппаратов, например, для сетей Петри с остановкой таймеров.
- Пополнение библиотеки моделей компонентов МВС новыми моделями, например, моделями средств взаимодействия задач в рамках одного раздела и моделями компонентов используемых в МВС сетей передачи данных.
- Интеграция разработанной инструментальной системы с САПР МВС, отличными от САПР «Планировщик ИМА», а также с другими средствами анализа конфигураций вычислительных систем реального времени (пример выполнения такой интеграции описан в работе [137]).

Список литературы

1. Mixed Criticality in Control Systems / A. Crespo [et al.] // IFAC Proceedings Volumes. Elsevier publ. — 2014. — Vol. 47, no. 3. — P. 12261—12271.
2. Парамонов П. П., Жаринов И. О. Интегрированные бортовые вычислительные системы: обзор современного состояния и анализ перспектив развития в авиационном приборостроении // Научно-технический вестник информационных технологий, механики и оптики. — 2013. — № 2. — С. 1—17.
3. Sysgo. Emdebbing innovations. Industry Solutions [Электронный ресурс]. — URL: https://www.sysgo.com/solutions/industry-solutions (дата обр. 15.12.2020).
4. Beckert M. Scheduling Mechanisms for Efficient and Safe Automotive Systems Integration: Thesis. — Braunschweig, Germany : Technische Universitat Braunschweig, 2019. — 204 p.
5. Fang H., Obermaisser R. Execution Environment for Mixed-Criticality Train Applications Based on an Integrated Architecture // 2017 International Conference on Promising Electronic Technologies (ICPET). — 2017. — P. 1—7.
6. DECOS: an Integrated Time-Triggered Architecture / R. Obermaisser [et al.] // e & i Elektrotechnik und Informationstechnik. — 2006. — Vol. 123, no. 3. — P. 83—95.
7. Федосов Е. А., Косьянчук В. В., Сельвесюк Н. И. Интегрированная модульная авионика // Радиоэлектронные технологии. — 2015. — № 1. — С. 66—71.
8. Liu C. L., Layland J. W. Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment // Journal of the ACM (JACM). — 1973. — Vol. 20, no. 1. — P. 46—61.
9. Applying New Scheduling Theory to Static Priority Preemptive Scheduling / N. Audsley [et al.] // Software Engineering Journal. — 1993. — Vol. 8, no. 5. — P. 284—292.
10. Tindell K., Clark J. Holistic Schedulability Analysis for Distributed Hard Real-Time Systems // Microprocessing and microprogramming. — 1994. — Vol. 40, no. 2/3. — P. 117—134.
11. Baruah S. K., Burns A., Davis R. I. Response-Time Analysis for Mixed Criticality Systems // 2011 32nd Real-Time Systems Symposium (RTSS). — 2011. — P. 34—43.
12. Timing Analysis of Mixed-Criticality Hard Real-Time Applications Implemented on Distributed Partitioned Architectures / S. O. Marinescu [et al.] // 2012 IEEE 17th Conference on Emerging Technologies & Factory Automation (ETFA). — 2012. — P. 1—4.
13. Schedulability Analysis of Hierarchical Systems with Arbitrary Scheduling in the Global Level / A. Guasque [et al.] // Proc. CESCIT. — 2015. — Vol. 48, no. 10. — P. 264—269.
14. Response-Time Analysis in Hierarchically-Scheduled Time-Partitioned Distributed Systems / J. C. Palencia [et al.] // IEEE Transactions on Parallel and Distributed Systems. — 2017. — Vol. 28, no. 7. — P. 2017—2030.

15. Wind River VxWorks 653 Platform 2.4 and 2.5. Product Note [Электронный ресурс]. — 2017. — URL: http://www.windriver.com/products/product-notes/vxworks-653-product-note.pdf (дата обр. 15.12.2020).

16. Годунов А. Н., Солдатов В. А. Спецификация ARINC 653 и ее реализация и операционной системе реального времени Багет 3 // Программная инженерия. — 2015. — № 6. — С. 3—17.

17. Macariu G., Cretu V. Timed Automata Model for Component-Based Real-Time Systems // 17th IEEE International Conference and Workshops on Engineering of Computer Based Systems. — 2010. — P. 121—130.

18. Глонина А. Б. Программное средство моделирования модульных вычислительных систем для проверки допустимости их конфигураций // Программные продукты и системы. — 2017. — Т. 30, № 4. — С. 574—582.

19. Balashov V. V., Balakhanov V. A., Kostenko V. A. Scheduling of Computational Tasks in Switched Network-Based IMA Systems // International Conference on Engineering and Applied Sciences Optimization. — Athens, Greece : National Technical University of Athens (NTUA), 2014. — P. 1001—1014.

20. Studying on ARINC653 Partition Run-time Scheduling and Simulation / D. Wang [et al.] // Proceedings of World Academy of Science, Engineering and Technology (WASET). Vol. 71. — 2012. — P. 1583—1587.

21. DYANA: HLA-based Distributed Real-time Embedded Systems Simulation Tool / V. A. Antonenko [et al.] // Proceedings of the 2013 Winter Simulation Conference: Simulation: Making Decisions in a Complex World. — 2013. — P. 4012—4013.

22. Smeliansky R. L., Bakhmurov A. G., Kapitonova A. P. Dyana: An Evironment for Embedded System Design and Analysis // Proceedings of the 32nd Annual Simulation Symposium. — San Diego, California, USA : IEEE Computer Society Press, 1999. — P. 50—57.

23. Cheddar: an open-source real-time schedulability tool/scheduling simulator [Электронный ресурс]. — URL: http://beru.univ-brest.fr/~singhoff/cheddar/ (дата обр. 15.12.2020).

24. HSSim: an Extensible Interoperable Object-Oriented n-Level Hierarchical Scheduling Simulator [Электронный ресурс]. — URL: https://github.com/jpgcc/hssim (дата обр. 15.12.2020).

25. MAST: Modeling and Analysis Suite for Real-Time Applications [Электронный ресурс]. — URL: https://mast.unican.es/ (дата обр. 15.12.2020).

26. Инструментальные средства проектирования систем интегрированной модульной авионики / Д. В. Буздалов [и др.] // Труды Института системного программирования РАН. — 2014. — Т. 26, № 1. — С. 201—230.

27. Система автоматизированного проектирования «Планировщик задач интегрированной модульной авионики боевых комплексов»: свид. о гос. регистрации программы для ЭВМ 2017611547 РФ / В. Балаханов [и др.]. — ФИПС, 2017.

28. Глонина А. Б. Инструментальная система проверки выполнения ограничений реального времени для конфигураций модульных вычислительных систем // Вестник Московского университета. Серия 15: Вычислительная математика и кибернетика. — Москва, 2020. — № 3. — С. 16—29.

29. Глонина А. Б., Балашов В. В. О корректности моделирования модульных вычислительных систем реального времени с помощью сетей временных автоматов // Моделирование и анализ информационных систем. — Ярославль, 2018. — Т. 25, № 2. — С. 174—192.

30. Глонина А. Б. Обобщенная модель функционирования модульных вычислительных систем реального времени для проверки допустимости конфигураций таких систем // Вестник Южно-Уральского государственного университета. Серия «Вычислительная математика и информатика». — 2017. — Т. 6, № 4. — С. 43—59.

31. Глонина А. Б. Программное средство моделирования модульных вычислительных систем для проверки допустимости их конфигураций // Программные продукты и системы. — 2017. — Т. 30, № 4. — С. 574—582.

32. Glonina A., Bahmurov A. Stopwatch Automata-Based Model for Efficient Schedulability Analysis of Modular Computer Systems // Parallel Computing Technologies (PaCT). Vol. 10421. — Cham, Switzerland : Springer, 2017. — P. 289—300. — (Lecture Notes in Computer Science).

33. Глонина А. Б. Масштабирование результатов верификации моделей компонентов модульных вычислительных систем // «Тихоновские чтения 2019»: научная конференция. Тезисы докладов. — Москва : МАКС Пресс, 2019. — С. 34.

34. Глонина А. Б. Обобщенная модель функционирования модульных вычислительных систем реального времени для проверки допустимости конфигураций таких систем // Суперкомпьютерные дни в России: Труды международной конференции (25–26 сентября 2017 г.) — Москва : Издательство МГУ, 2017. — С. 800—814.

35. Глонина А. Б., Балашов В. В., Бахмуров А. Г. Подход к использованию имитационного моделирования при решении задач синтеза и планирования в модульных вычислительных системах // Программные системы и инструменты. Тематический сборник. Т. 15. — Москва : Издательский отдел факультета ВМК МГУ, 2014. — С. 116—136.

36. Глонина А. Б. Обобщенная модель функционирования модульных вычислительных систем для проверки выполнения ограничений реального времени // «Тихоновские чтения 2018»: научная конференция. Тезисы докладов. — Москва : МАКС Пресс, 2018. — С. 93.

37. Глонина А. Б. Программное средство моделирования модульных вычислительных систем для проверки ограничений реального времени // «Ломоносовские чтения 2018»: научная конференция. Тезисы докладов. — Москва : МАКС Пресс, 2018. — С. 42.

38. Глонина А. Б., Бахмуров А. Г. Математическая модель функционирования модульных ВС РВ для проверки допустимости конфигураций таких систем // «Ломоносовские чтения 2017»: научная конференция. Тезисы докладов. — Москва : МАКС Пресс, 2017. — С. 74.

39. Глонина А. Б. Автоматическое построение имитационных моделей модульных ИУС РВ // «Ломоносовские чтения 2016»: научная конференция. Тезисы докладов. — Москва : Издательский отдел факультета ВМК МГУ, 2016. — С. 99—100.

40. Глонина А. Б., Балашов В. В. Использование автоматически построенных имитационных моделей при проектировании ИУС РВ с архитектурой ИМА // VIII Московская международная конференция по исследованию операций (ORM2016): Москва, 17–22 октября 2016 г. Т. 2. — Москва : ФИЦ ИУ РАН, 2016. — С. 168—169.

41. Глонина А. Б. Применение имитационного моделирования при решении задач синтеза и планирования для модульных вычислительных систем // Сборник тезисов XXII Международной научной конференции студентов, аспирантов и молодых ученых «Ломоносов-2015», секция «Вычислительная Математика и Кибернетика». — Москва : Издательский отдел факультета ВМК МГУ, 2015. — С. 93—94.

42. Глонина А. Б. Инструментальная система проверки выполнения ограничений реального времени для конфигураций модульных вычислительных систем: свид. о гос. регистрации программы для ЭВМ 2020611082 РФ. — ФИПС, 2020.

43. Aircraft Data Network Part 7. Avionics Full Duplex Switched Ethernet (AFDX) Network. ARINC Specification 664P7. — Annapolis, USA : Aeronautical Radio, 2005. — 132 p.

44. Осипов Ю. С., Першин А. С., Пустовой Ю. В. Способ передачи информации в реальном времени с использованием локальных сетей ограниченного размера на базе модификации протокола FC-AE-ASM: пат. 2536659 РФ. — Бюл. № 36, 2013.

45. Avionics Application Software Standard Interface. ARINC Specification 653. — Annapolis, USA : Aeronautical Radio, 1997. — 105 p.

46. Delange J., Lec L. POK, an ARINC653-Compliant Operating System Released under the BSD License // 13th Real-Time Linux Workshop. Vol. 10. — 2011. — P. 1—13.

47. Marouf M., Sorel Y. Scheduling Non-Preemptive Hard Real-Time Tasks with Strict Periods // 2011 IEEE 16th Conference on Emerging Technologies & Factory Automation (ETFA). — 2011. — P. 1—8.

48. Третьяков А. В. Автоматизация составления расписаний для систем реального времени. — 2012.

49. Work in Progress Paper: Pessimism Analysis of Network Calculus Approach on AFDX Networks / A. Soni [et al.] // 12th IEEE International Symposium on Industrial Embedded Systems (SIES). — 2017. — P. 1—4.

50. Платформа интегрированной модульной авионики: пат. 2413280 РФ / К. А. Егоров [и др.]. — Бюл. № 6, 2011.

51. Платформа интегрированной модульной авионики боевых комплексов: пат. 2595507 РФ / А. С. Першин [и др.]. — Бюл. № 24, 2016.

52. Конкурсная работа на конкурс «Авиастроитель года» Союза авиастроителей России. Бортовая цифровая вычислительная машина на основе принципов интегрированной модульной авионики боевых комплексов для информационно-управляющей системы самолета Су-57. [Электронный ресурс]. — 2017. — URL: http://www.aviationunion.ru/Files/Nom_7_GRPZ.PDF (дата обр. 15.12.2020).

53. Роль и место бортового оборудования воздушных судов на современном этапе развития авиации. Часть 2. [Электронный ресурс]. — 2014. — URL: http://www.modern-avionics.ru/analytics/2014/modern-role-of-avionics-aircraft/part-2 (дата обр. 15.12.2020).

54. Федосов Е. А. Основные направления формирования научно-технического задела в области бортового оборудования перспективных воздушных судов [Электронный ресурс]. — 2017. — URL: http://www.modern-avionics.ru/Files/01-GosNIIAS-Fedosov-20.07.2017.pdf (дата обр. 15.12.2020).

55. Модульный бортовый комплекс средств цифровой радиосвязи: пат. 2514098 РФ / А. В. Комяков [и др.]. — Бюл. № 12, 2014.

56. Интегрированный комплекс бортового оборудования разнородной архитектуры: пат. 2592193 РФ / О. Ф. Демченко [и др.]. — Бюл. № 20, 2016.

57. Интегрированная вычислительная система самолета МС-21: пат. 2667040 РФ / А. С. Баранов [и др.]. — Бюл. № 26, 2018.

58. Техническое описание ОС РВ «БАГРОС-4000» [Электронный ресурс]. — URL: https://www.sukhoi.org/bagros/ (дата обр. 15.12.2020).

59. Mallachiev K., Pakulin N., Khoroshilov A. Design and architecture of real-time operating system // Proceedings of the Institute for System Programming. — 2016. — Vol. 28, no. 2. — P. 181—192.

60. Колотилов Е. Д., Данилин П. Е., Стариченков Д. А. Технология проведения предварительных испытаний ФПО в условиях имитационной среды // Труды МИЭА. Навигация и управление летательными аппаратами. Т. 17. — Москва : Московский институт электромеханики и автоматики, 2017. — С. 13—22.

61. Park M. Non-preemptive Fixed Priority Scheduling of Hard Real-Time Periodic Tasks // 2007 International Conference on Computational Science. — 2007. — P. 881—888.

62. Francu C. A. Real-Time Scheduling for Java: Thesis. — Cambridge, Massachusetts, USA : Massachusetts Institute of Technology, 2002. — 59 p.

63. Model-based Framework for Schedulability Analysis Using Uppaal 4.1 / A. David [et al.] // Model-Based Design for Embedded Systems. — Boca Raton, Florida, USA : CRC Press, 2009. — P. 93—119.

64. Zhou T., Xionq H., Zhang Z. Hierarchical Resource Allocation for Integrated Modular Avionics Systems // Journal of Systems Engineering and Electronics. — 2011. — Vol. 22, no. 5. — P. 780—787.

65. Zhang C., Xiao J. Modeling and Optimization in Distributed Integrated Modular Avionics // 2013 IEEE/AIAA 32nd Digital Avionics Systems Conference (DASC). — 2013. — 2E11—2E112.

66. Challenges in Building Fault-Tolerant Flight Control System for a Civil Aircraft. / M. Sghairi [et al.] // IAENG International Journal of Computer Science. — 2008. — Vol. 35, no. 4.

67. Al Sheikh A., Brun O., Hladik P.-E. Partition Scheduling on an IMA Platform with Strict Periodicity and Communication Delays // 18th International Conference on Real-Time and Network Systems. — Toulouse, France, 2010. — P. 179—188.

68. Annighöfer B., Thielecke F. A Systems Architecting Framework for Optimal Distributed Integrated Modular Avionics Architectures // CEAS Aeronautical Journal. — 2015. — Vol. 6, no. 3. — P. 485—496.

69. Scheduling Based on Interruption Analysis and PSO for Strictly Periodic and Preemptive Partitions in Integrated Modular Avionics / H. Lu [et al.] // IEEE Access. — 2018. — Vol. 6. — P. 13523—13540.

70. TS-Preemption Threshold and Priority Optimization for the Process Scheduling in Integrated Modular Avionics / Q. Zhou [et al.] // Bio-inspired Computing: Theories and Applications. — 2017. — P. 9—23.

71. Chen J., Du C., Han P. Scheduling Independent Partitions in Integrated Modular Avionics Systems // PloS one. — 2016. — Vol. 11, no. 12. — P. 1—18.

72. Craveiro J. P., Silveira R. O., Rufino J. hsSim: an Extensible Interoperable Object-Oriented n-Level Hierarchical Scheduling Simulator // Proceedings of the 3rd International Workshop on Analysis Tools and Methodologies for Embedded and Real-time Systems (WATERS). — Pisa, Italy, 2012. — P. 9—14.

73. Smelyansky R. L. Model of Distributed Computing System Operation with Time // Programming and Computer Software. — 2013. — Vol. 39, no. 5. — P. 233—241.

74. The SMART Project: Multi-Agent Scheduling Simulation of Real-time Architectures / P. Dissaux [et al.] // Embedded Real Time Software and Systems. — Toulouse, France, 2014. — P. 1—9.

75. Formal Verification Method for Configuration of Integrated Modular Avionics System Using MARTE / L. Wang [et al.] // International Journal of Aerospace Engineering. — 2018. — Vol. 2018. — P. 1—22.

76. Костенко В. А. Задача построения расписания при совместном проектировании аппаратных и программных средств // Программирование. — 2002. — № 3. — С. 64—80.

77. Leung J. Y.-T., Merrill M. A Note on Preemptive Scheduling of Periodic, Real-Time Tasks // Information processing letters. — 1980. — Vol. 11, no. 3. — P. 115—118.

78. Jiang Y., Cheng A. M., Zou X. Schedulability Analysis for Real-Time P-FRP Tasks under Fixed Priority Scheduling // 2015 IEEE 21st International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA). — 2015. — P. 31—40.

79. Henzinger T. A. The Theory of Hybrid Automata // 11th Annual IEEE Symposium on Logic in Computer Science. — 1996. — P. 278—292.

80. Программные средства моделирования непрерывно-дискретных систем / В. М. Глушков [и др.]. — Киев : Наукова думка, 1975. — 152 с.

81. Бусленко Н. П. Моделирование сложных систем. — Москва : Наука, 1978. — 400 с.

82. Парийская Е. Ю. Сравнительный анализ математических моделей и подходов к моделированию и анализу непрерывно–дискретных систем // Дифференциальные уравнения и процессы управления. — 1997. — № 1. — С. 92—120.

83. Alur R., Dill D. The Theory of Timed Automata // Real-Time: Theory in Practice. — Berlin, Heidelberg, Germany : Springer, 1992. — P. 45—73.

84. Alur R., Courcoubetis C., Dill D. Model-Checking for Real-Time Systems // Fifth Annual IEEE Symposium on Logic in Computer Science. — 1990. — P. 414—425.

85. Symbolic Model Checking for Real-time Systems / T. A. Henzinger [et al.] // Information and Computation. — 1994. — Vol. 111, no. 2. — P. 193—244.

86. Bengtsson J., Yi W. Timed Automata: Semantics, Algorithms and Tools // Lectures on Concurrency and Petri Nets: Advances in Petri Nets. — Berlin, Heidelberg, Germany : Springer, 2004. — P. 87—124.

87. Milner R. A Calculus of Communicating Systems. — Secaucus, NJ, USA : Springer-Verlag: 1982. — 260 p.

88. Clarke E. M., Emerson E. A. Design and Synthesis of Synchronization Skeletons Using Branching Time Temporal Logic // Logics of Programs. — Berlin, Heidelberg, Germany : Springer, 1982. — P. 52—71.

89. A Formal Model of the Ada Ravenscar Tasking Profile. Protected Objects / K. Lundqvist [et al.] // Reliable Software Technologies — Ada-Europe 99. — Berlin, Heidelberg, Germany : Springer, 1999. — P. 12—25.

90. UPPAAL Home [Электронный ресурс]. — URL: http://uppaal.org/ (дата обр. 15.12.2020).

91. IMITATOR − Parameter Synthesis for Real-Time Systems [Электронный ресурс]. — URL: https://www.imitator.fr/ (дата обр. 15.12.2020).

92. André É. Observer Patterns for Real-Time Systems // Proceedings of the 2013 18th International Conference on Engineering of Complex Computer Systems. — 2013. — P. 125—134.

93. Fersman E., Pettersson P., Yi W. Timed Automata with Asynchronous Processes: Schedulability and Decidability // Tools and Algorithms for the Construction and Analysis of Systems. — Berlin, Heidelberg, Germany : Springer, 2002. — P. 67—82.

94. Cassez F., Larsen K. The Impressive Power of Stopwatches // CONCUR 2000 — Concurrency Theory. — Berlin, Heidelberg, Germany : Springer, 2000. — P. 138—152.

95. Krishna S. N., Manasa L., Trivedi A. What's Decidable About Recursive Hybrid Automata? // Proceedings of the 18th International Conference on Hybrid Systems: Computation and Control (HSCC '15). — Seattle, Washington : ACM, 2015. — P. 31—40.

96. On Reachability for Hybrid Automata over Bounded Time / T. Brihaye [et al.] // Automata, Languages and Programming. — Berlin, Heidelberg, Germany : Springer, 2011. — P. 416—427.

97. Abdeddaïm Y., Maler O. Preemptive Job-Shop Scheduling Using Stopwatch Automata // Tools and Algorithms for the Construction and Analysis of Systems. — Berlin, Heidelberg, Germany : Springer, 2002. — P. 113—126.

98. Schedulability Analysis Using Two Clocks / E. Fersman [et al.] // Tools and Algorithms for the Construction and Analysis of Systems. — Berlin, Heidelberg, Germany : Springer, 2003. — P. 224—239.

99. Krčál P., Yi W. Decidable and Undecidable Problems in Schedulability Analysis Using Timed Automata // Tools and Algorithms for the Construction and Analysis of Systems. — Berlin, Heidelberg, Germany : Springer, 2004. — P. 236—250.

100. Питерсон Д. Теория сетей Петри и моделирование систем. — Москва : Мир, 1984. — 264 с.

101. Bobbio A. System Modelling with Petri Nets // Systems Reliability Assessment. — Dordrecht, Netherlands : Springer, 1990. — P. 103—143.

102. Merlin P., Farber D. Recoverability of Communication Protocols — Implications of a Theoretical Study // IEEE Transactions on Communications. — 1976. — Vol. 24, no. 9. — P. 1036—1043.

103. Walter B. Timed Petri-Nets for Modelling and Analyzing Protocols with Real-Time Characteristics. // Protocol Specification, Testing, and Verification. — 1983. — P. 149—159.

104. Akshay S., Genest B., Hélouët L. Decidable Classes of Unbounded Petri Nets with Time and Urgency // Application and Theory of Petri Nets and Concurrency. — Cham, Switzerland : Springer, 2016. — P. 301—322.

105. The TINA Toolbox Home Page — TIme petri Net Analyzer [Электронный ресурс]. — URL: http://projects.laas.fr/tina/ (дата обр. 15.12.2020).

106. Roméo — A tool for Time Petri Nets Analysis [Электронный ресурс]. — URL: http://romeo.rts-software.org/ (дата обр. 15.12.2020).

107. Cassez F., Roux O. H. Structural Translation from Time Petri Nets to Timed Automata // Journal of Systems and Software. — 2006. — Vol. 79, no. 10. — P. 1456—1468. — Architecting Dependable Systems.

108. Byg J., Jørgensen K. Y., Srba J. TAPAAL: Editor, Simulator and Verifier of Timed-Arc Petri Nets // Automated Technology for Verification and Analysis. — Berlin, Heidelberg, Germany : Springer, 2009. — P. 84—89.

109. Delgadillo G. M., Llano S. P. Scheduling Application Using Petri Nets: A Case Study: Intergráficas S.A. // Proceedings of 19th International Conference on Production Research, Valparaiso, Chile. — 2006. — P. 1—6.

110. Allahham A., Alla H. Post and Pre-Initialized Stopwatch Petri Nets: Formal Semantics and State Space Computation // Nonlinear Analysis: Hybrid Systems. — 2008. — Vol. 2, no. 4. — P. 1175—1186.

111. Миронов А. Теория процессов. — Переславль-Залесский : Университет города Переславля, 2008. — 345 с.

112. Захаров В. А. Модели последовательных и параллельных вычислений [Электронный ресурс]. — 2017. — URL: http://mk.cs.msu.ru/index.php/Модели_последовательных_и_параллельных_вычислений (дата обр. 15.12.2020).

113. Baeten J. C. M., Bergstra J. A. Real-Time Process Algebra // Formal Aspects of Computing. — 1991. — Vol. 3, no. 2. — P. 142—188.

114. Yi W. CCS + Time = an Interleaving Model for Real-Time Systems // Automata, Languages and Programming. — Berlin, Heidelberg, Germany : Springer, 1991. — P. 217—228.

115. Reed G. M., Roscoe A. W. A Timed Model for Communicating Sequential Processes // Theoretical Computer Science. — 1988. — Vol. 58, no. 1—3. — P. 249—261.

116. Model Checking Process Algebra of Communicating Resources for Real-Time Systems / A. J. Boudjadar [et al.] // 2014 26th Euromicro Conference on Real-Time Systems. — 2014. — P. 51—60.

117. Смелянский Р. Л. Анализ производительности многопроцессорных систем на основе инвариантности поведения программ : дис. ... д-ра физ.-мат. наук : 05.13.11. — Москва : МГУ, 1991. — 309 с.

118. DR TESY. Methods and Tools for Distributed Real Time Embedded Systems, Design and Analysis. Report on Concepts for Integration. — Berlin, Germany : GMD — Forschungszentrum Informationstechnik GmbH Institute for Computer Systems, Software Technology (FIRST), 2000. — 53 p.

119. DO-297: Integrated Modular Avionics (IMA) Development Guidance and Certification Considerations. — Washington DC, USA : Radio Technical Commission for Aeronautics, Inc. (RTCA), 2005. — 137 p.

120. Resource Scheduling in Dependable Integrated Modular Avionics / Y.-H. Lee [et al.] // International Conference on Dependable Systems and Networks (DSN 2000). — 2000. — P. 14—23.

121. Pnueli A., Manna Z. The Temporal Logic of Reactive and Concurrent Systems. — New York, USA : Springer, 1992. — 427 p.

122. Emerson E. A., Halpern J. Y. «Sometimes» and «Not Never» Revisited: on Branching Versus Linear Time Temporal Logic // Journal of the ACM (JACM). — 1986. — Vol. 33, no. 1. — P. 151—178.

123. Koymans R. Specifying Real-Time Properties with Metric Temporal Logic // Real-Time Systems. — 1990. — Vol. 2, no. 4. — P. 255—299.

124. CTRL: Extension of CTL with Regular Expressions and Fairness Operators to Verify Genetic Regulatory Networks / R. Mateescu [et al.] // Theoretical Computer Science. — 2011. — Vol. 412, no. 26. — P. 2854—2883.

125. Beer I., Ben-David S., Landver A. On-the-Fly Model Checking of RCTL Formulas // Computer Aided Verification. — Berlin, Heidelberg, Germany : Springer, 1998. — P. 184—194.

126. Leucker M., Sánchez C. Regular Linear Temporal Logic // Theoretical Aspects of Computing — ICTAC 2007. — Berlin, Heidelberg, Germany : Springer, 2007. — P. 291—305.

127. Daniels D. Safety Aspects of a Landing Gear System // Developments in Risk-Based Approaches to Safety. — London, UK : Springer, 2006. — P. 199—213.

128. A Deadline-Floor Inheritance Protocol for EDF Scheduled Embedded Real-Time Systems with Resource Sharing / A. Burns [et al.] // IEEE Transactions on Computers. — 2015. — Vol. 64, no. 5. — P. 1241—1253.

129. Xu J., Parnas D. L. Scheduling Processes with Release Times, Deadlines, Precedence and Exclusion Relations // IEEE Transactions on Software Engineering. — 1990. — Vol. 16, no. 3. — P. 360—369.

130. Adaptive Task Automata with Earliest-Deadline-First Scheduling / L. Hatvani [et al.] // Electronic Communications of the EASST. — 2014. — Vol. 70. — P. 1—15.

131. Apt K. R., Kozen D. C. Limits for Automatic Verification of Finite-State Concurrent Systems // Information Processing Letters. — 1986. — Vol. 22, no. 6. — P. 307—309.

132. Cheetah3, the Python-Powered Template Engine [Электронный ресурс]. — URL: http : / / cheetahtemplate.org/ (дата обр. 15.12.2020).

133. GCC, the GNU Compiler Collection [Электронный ресурс]. — URL: https : / / gcc . gnu . org/ (дата обр. 15.12.2020).

134. Tool System and Algorithms for Scheduling of Computations in Integrated Modular Onboard Embedded Systems / V. Balashov [et al.] // IFAC Proceedings Volumes (IFAC-PapersOnline). — 2016. — Vol. 49, no. 25. — P. 505—510.

135. Balashov V., Antipina E. Distribution of Workload in IMA Systems by Solving a Modified Multiple Knapsack Problem // Proc. 6th International Conference on Engineering Optimization. — Cham, Switzerland : Springer, 2018. — P. 1166—1177.

136. Кремер Н. Ш., Путко Б. А. Эконометрика. — Москва : ЮНИТИ-ДАНА, 2002. — 311 с.

137. Гонопольский М. Г., Глонина А. Б. Алгоритм оценки максимального времени отклика задач в многопроцессорных системах с интервальной неопределенностью длительности выполнения работ // Моделирование и анализ информационных систем. — Ярославль, 2020. — Т. 27, № 2. — С. 218—233.

138. André É. What's Decidable about Parametric Timed Automata? // International Journal on Software Tools for Technology Transfer. — 2019. — Vol. 21, no. 2. — P. 203—219.

139. Захаров В. А. Математические методы верификации схем и программ [Электронный ресурс]. — 2018. — URL: http : / / mk . cs . msu . ru / index . php / Математические _ методы _ верификации_схем_и_программ (дата обр. 15.12.2020).

140. Карпов Ю. Г. Model Checking: верификация параллельных и распределенных программных систем. — Санкт-Петербург : БХВ-Петербург, 2010. — 550 с.

141. Alur R., Dill D. L. A Theory of Timed Automata // Theoretical Computer Science. — 1994. — Vol. 126, no. 2. — P. 183—235.

Приложение А

Разработанные модели компонентов МВС

А.1. Модель планировщика ядра

Параметризованный автомат, моделирующий планировщик ядра, представлен на рисунке А.1.

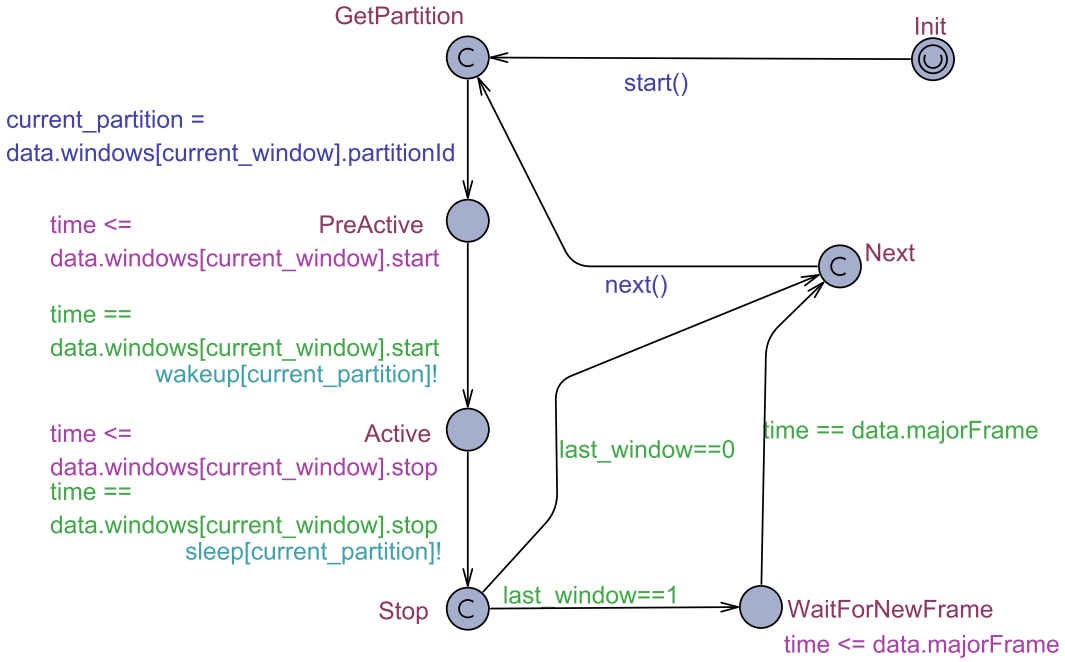

Рисунок А.1 — Модель планировщика ядра

Согласно разделу 2.4, автомат работает с массивами каналов $sleep[]$ и $wakeup[]$.[1] Количество элементов в каждом из этих массивов равно числу разделов, привязанных к соответствующему ядру, и содержится в описании конфигурации. Автомат работает с параметрами $data.majorFrame$, $data.numOfWindows$ и с массивами параметров $data.windows[].start$, $data.windows[].stop$, $data.windows[].partitionId$ (размер всех массивов задает параметр $numOfWindows$). Значения этих параметров инициализируются значениями соответствующих характеристик конфигурации (см. 2.4).

У автомата имеются внутренние переменные $current\_window$ (текущий номер окна), $current\_partition$ (текущий номер раздела), $last\_window$ (флаг обработки последнего окна в расписании). Начальные значения всех переменных равны 0. Также в данном автомате используется таймер $time$.

[1] В этом приложении массивы обозначаются записями вида $sleep[]$, а элементы массивов — $sleep[i]$. При этом, например, обозначение $sleep[i]$ соответствует обозначению $sleep_i$ обобщенной модели.

Начальной локацией автомата является локация $Init$. В данной локации запрещено продвижение времени. Из локации существует один переход — в локацию $GetPartition$. При выполнении этого перехода выполняется функция $start()$, в которой переменной $last\_window$ присваивается значение 1, если расписание окон содержит всего одно окно. Иначе значение этой переменной остается равно 0.

Локация $GetPartition$ соответствует определению номера раздела, соответствующего очередному окну. В данной локации запрещено продвижение времени. Из локации существует один переход — в локацию $PreActive$. При выполнении этого перехода переменной $current\_partition$ присваивается значение, равное номеру раздела, соответствующего окну с номером $current\_window$.

Локация $PreActive$ соответствует ожиданию открытия очередного окна: инвариант данной локации имеет вид $time <= data.windows[current\_window].start$, где $data.windows[current\_window].start$ — время открытия текущего окна согласно расписанию. Как только значение таймера $time$ становится равно времени открытия очередного окна, выполняется переход в локацию $Active$ и при этом выполняется посылка сигнала по каналу $wakeup[current\_partition]$ (для активации планировщика работ соответствующего раздела).

Локация $Active$ соответствует выполнению работ некоторого раздела в рамках своего окна: инвариант данной локации имеет вид $time <= data.windows[current\_window].stop$, где $data.windows[current\_window].stop$ — время закрытия текущего окна согласно расписанию. Как только значение таймера $time$ становится равно времени закрытия текущего окна, выполняется переход в локацию $Stop$ и при этом выполняется посылка сигнала по каналу $sleep[current\_partition]$ (для деактивации планировщика работ соответствующего раздела).

Из локации $Stop$ осуществляется переход в локацию $Next$, если переменная $last\_window$ равна 0 и в локацию $WaitForNextFrame$, иначе. В локации $Stop$ запрещено продвижение времени.

Локация $Next$ соответствует переходу к следующему окну. В данной локации запрещено продвижение времени. Из локации существует один переход — в локацию $GetPartition$. При выполнении этого перехода выполняется функция $next()$, в которой переменной $current\_window$ присваивается значение 0 и обнуляется значение таймера $time$, если очередная отработка интервала планирования закончена; иначе значение переменной $current\_window$ увеличивается на 1. Также в функции $next()$ переменной $last\_window$ присваивается значение 1, если очередное окно является последним в расписании окон, и 0 — иначе.

Локация $WaitForNextFrame$ соответствует ожиданию наступления очередного интервала планирования и имеет инвариант $time <= data.majorFrame$. Как только таймера $time$ становится равно длительности интервала планирования, выполняется переход в локацию $Next$.

Локации $Init$, $GetPartition$, $Next$ и $PreActive$ соответствуют локациям A, B, C и D из определения класса автоматов У7 на странице 180, функции $start()$ и $next()$ имеют описанный в этом определении вид. Все условия определения класса У7 выполнены для построенного автомата, и этот автомат принадлежит классу У7.

Покажем, что описанная модель является периодической с периодом $data.majorFrame$ в терминах приложения Б. Единственный переход, в действиях которого содержится действие $time = 0$ — переход из локации $Next$ в локацию $GetPartition$ с выполнением функции $next()$. При этом, согласно ограничениям на вид функции $next()$, введенным при определении класса У7, действие $time = 0$ выполняется, *только* если истинно условие $current\_window == data.numOfWindows - 1$, что эквивалентно истинности условия $last\_window == 1$. Следовательно, при выполнении функции $next()$ действие $time = 0$ выполняется, *только* если переход в локацию $Next$ был выполнен из локации $WaitForNextFrame$. Поскольку в локации $Next$ запрещено продвижение времени, а условие перехода из локации $WaitForNextFrame$ в локацию $Next$ имеет вид $time == data.majorFrame$, то непосредственно перед выполнением перехода из локации $Next$ в локацию $GetPartition$ с выполнением действия $time = 0$ условие $time == data.majorFrame$ всегда тождественно истинно, что соответствует условию 1 периодичности автомата (см. стр. 162). Условие 2 выполняется по построению, выполнение условия 3 следует из ограничений на входные данные (расписание окон). Таким образом, модель является периодической с периодом $data.majorFrame$.

Количество временных параметров для данного автомата равно значению параметра $data.numOfWindows$, умноженному на 2. Таймеры с возможностью останова в данном автомате не используются. Количество индексных параметров для данного автомата равно значению параметра $data.numOfWindows$.

Таким образом, достаточные для верификации построенного автомата диапазоны значений параметров определяются согласно утверждениям 7, 8, 10 приложения Б.

А.2. Модели планировщиков работ разделов

А.2.1. Планировщик с фиксированными приоритетами и вытеснением

Параметризованный автомат, моделирующий планировщик с фиксированными приоритетами и вытеснением, представлен на рисунке А.2.

Согласно разделу 2.4, автомат работает с массивами переменных интерфейса $abs\_deadline[]$, $id[]$, $prio[]$, $is\_ready[]$. Размер этих массивов задает параметр $data.numOfTasks$ — число задач раздела. Переменные $abs\_deadline[]$ и $id[]$ в автомате не используются (они входят в интерфейс данного параметризованного автомата, т.к. входят в интерфейс соответствующего базового типа). Также этот автомат работает с каналами $ready$, $finished$, $sleep$, $wakeup$ и массивами каналов $exec[]$, $preempt[]$. Количество каналов в каждом из указанных массивов также равно параметру $numOfTasks$. Кроме того, у автомата имеется внутренняя переменная $current$, хранящая номер задачи, работа которой была поставлена на выполнение последней. Начальное значение этой переменной равно -1.

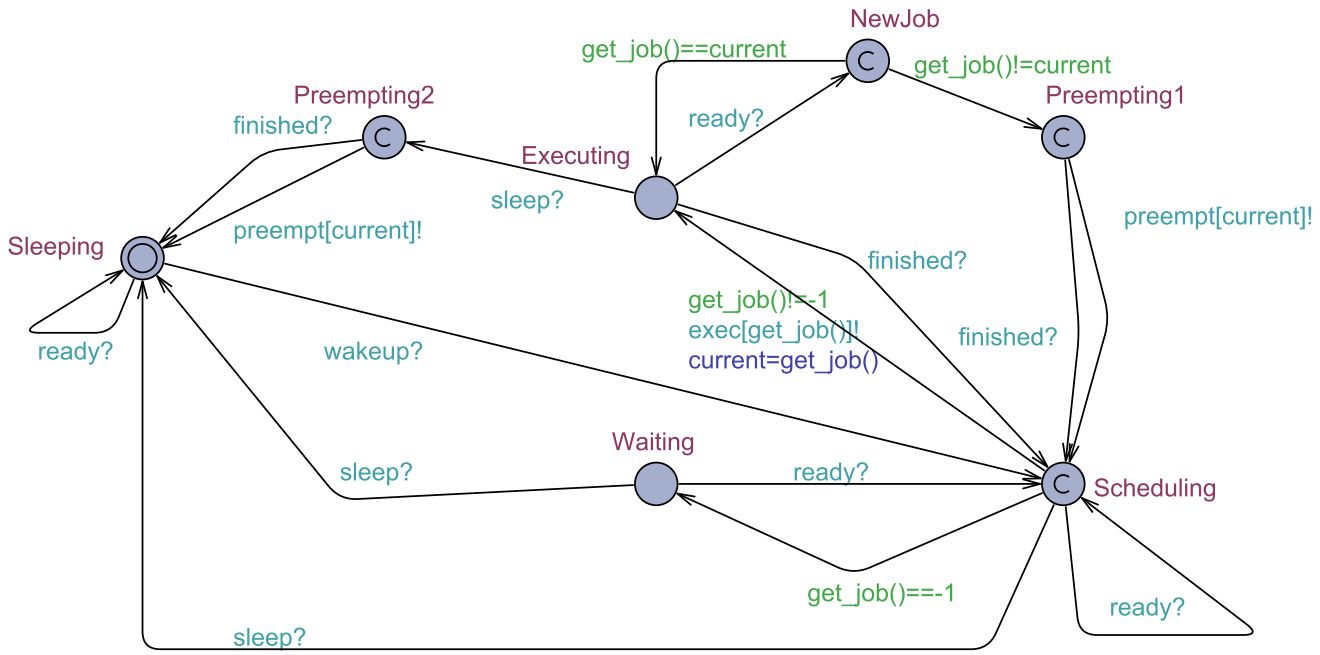

Рисунок А.2 — Модель планировщика с фиксированными приоритетами и вытеснением

Начальной локацией автомата является локация $Sleeping$. Эта локация должна быть текущей в моменты времени, соответствующие окнам других разделов. Синхронизации по каналам $ready$ (появление новой готовой работы) и $finished$ (завершение некоторой работы) не приводят к выполнению каких-либо содержательных действий, если текущей локацией является локация $Sleeping$. Синхронизация по каналу $wakeup$ (открытие окна раздела) приводит к переходу в локацию $Scheduling$.

В локации $Scheduling$ происходит выбор работы для постановки на выполнение (в т.ч. может быть принято решение не ставить на выполнение никакую работу). В этой локации запрещено продвижение времени, то есть принятие решения осуществляется мгновенно. Номер задачи, работа которой должна быть поставлена на выполнение, вычисляется с помощью функции $get\_job()$. Эта функция возвращает номер той задачи, работа которой готова и имеет наибольший приоритет среди всех готовых работ. Если готовых работ нет, то $get\_job()$ возвращает -1. Если некоторая работа должна быть поставлена на выполнение, то осуществляется изменение переменной $current$, синхронизация по соответствующему каналу из массива каналов $exec[]$ и переход в локацию $Executing$. Иначе осуществляется переход в локацию $Waiting$. Если в локации $Scheduling$ происходит синхронизация по каналу $sleep$ (закрытие окна раздела), то осуществляется переход в локацию $Sleeping$. Также в локации $Scheduling$ возможно выполнение синхронизации по каналу $ready$ (появление новой готовой работы), при этом текущая локация не изменяется. Согласно разделу 2.4 приоритет канала $ready$ больше, чем приоритет каналов массива $exec[]$, поэтому, если возможны синхронизации по нескольким каналам, то сначала будут выполнены все возможные синхронизации по каналам $ready$, а лишь затем — синхронизация по одному из каналов массива $exec[]$.

Локация $Waiting$ соответствует отсутствию готовых работ при открытом окне раздела. Если в этой локации происходит получение сигнала по каналу $sleep$ (закрытие окна раздела), то

выполняется переход в локацию $Sleeping$. Если происходит получение сигнала по каналу $ready$ (появление новой готовой работы), то выполняется переход в локацию $Scheduling$.

Локация $Executing$ соответствует выполнению некоторой работы. Если в этой локации происходит получение сигнала по каналу $finished$ (завершение некоторой работы), то выполняется переход в локацию $Scheduling$ для выбора новой работы для постановки на выполнение. Если в локации $Executing$ происходит получение сигнала по каналу $ready$ (появление новой готовой работы), то выполняется переход в локацию $NewJob$ для определения, должна ли быть вытеснена выполняющаяся работа. Если в локации $Executing$ происходит получение сигнала по каналу $sleep$ (закрытие окна раздела), то выполняется переход в локацию $Preempting2$ для вытеснения выполняющейся работы.

Локация $Preempting2$ соответствует состоянию планировщика раздела в момент закрытия окна этого раздела при наличии выполняющейся работы. В этой локации запрещено продвижение времени: либо мгновенно должен быть получен сигнал по каналу $finished$ (завершение выполняющейся работы), либо мгновенно должен быть отправлен сигнал по каналу $preempt[current]$ (вытеснение текущей работы). В обоих случаях из локации $Preempting2$ осуществляется переход в локацию $Sleeping$.

Локация $NewJob$ соответствует состоянию планировщика в момент появления новой готовой работы при наличии уже выполняющейся работы. В этом состоянии планировщик должен определить, нужно ли вытеснять выполняющуюся работу. В локации $NewJob$ запрещено продвижение времени, то есть решение должно быть принято мгновенно. Если значение, возвращаемое функцией $get\_job()$, совпадает со значением переменной $current$ (в данный момент времени, несмотря на появление новой готовой работы, должна выполняться та же работа, которая уже выполняется), то выполняется возврат в локацию $Executing$. Иначе выполняется переход в локацию $Preempting1$ для вытеснения выполняющейся работы.

Локация $Preempting1$ соответствует состоянию планировщика раздела в момент, когда выполняющаяся работа должна быть вытеснена. Эта локация аналогична локации $Preempting2$ с тем отличием, что из нее выполняются переходы в локацию $Executing$, а не $Sleeping$, т.к. после завершения или вытеснения текущей работы новая работа должна быть поставлена на выполнение.

Описанный автомат удовлетворяет условиям определения класса У6 на странице 165 и имеет одну функцию ($get\_job()$) и одну индексную переменную ($current$). Таким образом, достаточный для верификации построенного автомата диапазон значений параметра $data.numOfTasks$ определяется согласно утверждению 6 приложения Б.

А.2.2. Планировщик, работающий по стратегии EDF с вытеснением

Параметризованный автомат, моделирующий планировщик, работающий по стратегии EDF с вытеснением, аналогичен параметризованному автомату, моделирующему планировщик с фиксированными приоритетами и вытеснением. Имеются лишь следующие отличия:

– Переменные $prio[]$ в параметризованном автомате не используются, а используются переменные $abs\_deadline[]$ и $id[]$;
– Функция $get\_job()$ возвращает номер той задачи, работа которой готова и имеет наименьшую правую границу директивного интервала среди всех готовых работ. Если таких задач несколько, то функция возвращает минимальный из номеров задач, имеющих готовую работу с минимальным абсолютным директивным сроком.

Достаточные для верификации этого автомата диапазоны значений параметров определяются согласно утверждению 6 приложения Б.

А.2.3. Планировщик с фиксированными приоритетами без вытеснения

Параметризованный автомат, моделирующий планировщик с фиксированными приоритетами без вытеснения, приведен на рисунке А.3.

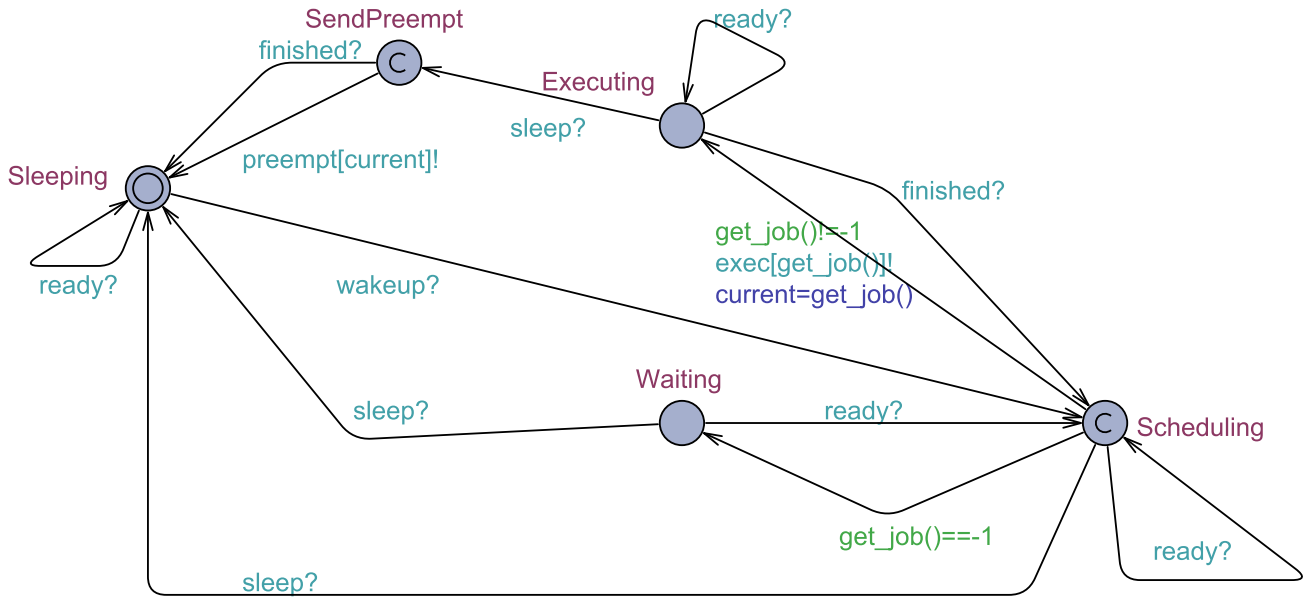

Рисунок А.3 — Модель планировщика с фиксированными приоритетами без вытеснения

Эта модель аналогична модели планировщика с фиксированными приоритетами и вытеснением. Отличие состоит лишь в том, что если автомат находится в локации $Executing$ и происходит получение сигнала по каналу $ready$, то текущая локация автомата не меняется. То есть появление новой готовой работы не приводит к каким-либо действиям в планировщике при наличии уже выполняющейся работы. Поэтому локации $NewJob$ и $Preempting1$ в модели планировщика с фиксированными приоритетами без вытеснения отсутствуют.

Достаточные для верификации этого автомата диапазоны значений параметров определяются согласно утверждению 6 приложения Б.

А.3. Модель функциональной задачи

Параметризованный автомат, моделирующий функциональную задачу приведен на рисунке А.4.

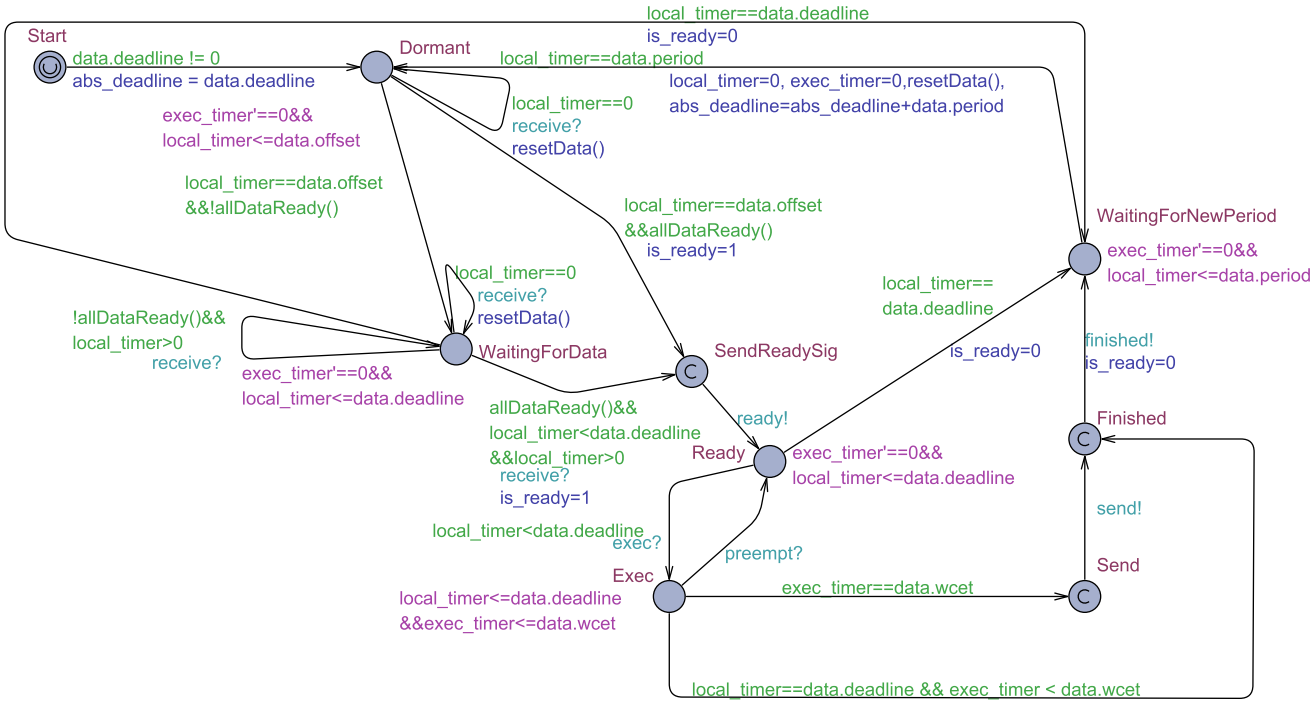

Рисунок А.4 — Модель функциональной задачи

Согласно разделу 2.4, автомат работает с параметрами `data.wcet`, `data.period`, `data.offset`, `data.deadline`, `data.numOfInputLinks`, задающими соответственно следующие характеристики задачи: WCET, период, смещение левой и правой границ директивных интервалов работ относительно начала периода, количество входных сообщений. Автомат работает с переменными интерфейса $abs\_dealine$, is_ready, $prio$, id, массивом переменных интерфейса $is\_data\_ready[]$ (количество переменных в массиве равно количеству задач, от которых данная задача синхронно зависит по данным). Также автомат работает с каналами $exec$, $preempt$, $ready$, $finished$, широковещательными каналами $send$, $receive$. Автомат использует два таймера: $local\_timer$ — для измерения времени, прошедшего с начала очередного периода, и $exec\_timer$ — для измерения чистого времени выполнения очередной работы на процессоре. Таймер $local\_timer$ активен во всех локациях автомата, таймер $exec\_timer$ активен только в локации $Exec$, соответствующей выполнению работы на вычислителе. В условиях некоторых переходов используется функция $allDataReady()$, возвращающая значение $true$, если все переменные $is\_data\_ready[]$ имеют истинное значение или задача не имеет зависимостей по данным, и $false$ иначе. В действиях, выполняемых при некоторых переходах, используется функция $ResetData()$, присваивающая всем переменным $is\_data\_ready[]$ значение $false$. Эта функция соответствует считыванию всех доставленных сообщений, и сразу после ее выполнения новых доставленных, но не прочитанных сообщений не имеется.

Начальной локацией автомата является локация $Start$. В данной локации запрещено продвижение времени. Назначение этой локации состоит в начальной инициализации переменной $abs\_deadline$. Решение об инициализации этой переменной внутри автомата, а не вне (при создании переменной), было принято исходя из того, что внутри автомата реализована логика изменения этой переменной. И, даже если при создании переменной ей было присвоено некоторое значение, то это значение будет перезаписано в соответствии с логикой функционирования автомата. Из локации $Start$ имеется один переход — в локацию $Dormant$.

Локация $Dormant$ соответствует промежутку времени между началом очередного периода и левой границей директивного интервала очередной работы. При переходе в эту локацию из других локаций таймер $local\_timer$ обнуляется. Если в локации $Dormant$ на границе периодов происходит получение сигнала по каналу $receive$ (соответствует получению сообщения), то считается, что полученное сообщение относится к работе, период которой заканчивается, и вызывается функция $resetData()$ (сообщение сбрасывается). Сообщение, полученное на границе периодов не может относиться к работе, период которой только начинается, т.к. в противном случае это означало бы, что работа-отправитель сообщения успела выполниться за 0 единиц времени. Когда значение таймера $local\_timer$ достигает левой границы директивного интервала работы, происходит переход из локации $Dormant$. Если $allDataReady()$ возвращает $true$ (т.е. либо к этому моменту получены все необходимые сообщения, либо задача не имеет зависимостей по данным), то происходит переход в локацию $SendReadySig$, иначе — в локацию $WaitingForData$.

Локация $WaitingForData$ соответствует ожиданию сообщений от работ-отправителей после достижения левой границы директивного интервала текущей работы. Аналогично локации $Dormant$, сообщения, полученные на границе периодов, в локации $WaitingForData$ сбрасываются. При получении очередного сообщения не на границе периодов (т.е. при получении сигнала по каналу $receive$) происходит проверка получения всех сообщений ($allDataReady()$). Если все сообщения получены, то происходит переход в локацию $SendReadySig$, иначе локация не изменяется. Автомат может находиться в локации $Dormant$ до достижения значением таймера $local\_timer$ правой границы директивного интервала текущей работы. Как только правая граница директивного интервала работы достигнута, происходит переход в локацию $WaitingForNewPeriod$ для ожидания появления новой работы моделируемой функциональной задачи.

Локация $SendReadySig$ соответствует состоянию функциональной задачи, в котором текущая работа задачи получила все необходимые сообщения от работ-отправителей, достигнута левая граница директивного интервала работы, но не достигнута правая граница этого интервала. В данной локации запрещено продвижение времени. Сразу после перехода в эту локацию, из нее происходит переход в локацию $Ready$ и при этом выполняется отправка сигнала по каналу $ready$.

Локация $Ready$ соответствует готовности очередной работы и ожиданию ее постановки на выполнение. Если в этой локации происходит получение сигнала по каналу $exec$ (постановка работы на выполнение), то выполняется переход в локацию $Exec$. Локация $Ready$ может являться текущей до тех пор, пока значение таймера $local\_timer$ не достигнет правой границы директивного интервала работы. Как только правая граница директивного интервала работы достигнута, происходит переход в локацию $WaitingForNewPeriod$.

Локация $Exec$ соответствует выполнению работы на вычислителе. Если в этой локации происходит получение сигнала по каналу $preempt$, то выполняется возврат в локацию $Ready$. Локация $Exec$ может являться текущей до тех пор, пока не наступит хотя бы одно из следующих событий: значение таймера $local\_timer$ достигнет правой границы директивного интервала работы; значение таймера $exec\_timer$ достигнет заданного для задачи значения WCET. Если значение таймера $local\_timer$ становится равным правой границе директивного интервала работы, то происходит переход в локацию $Finished$ для информирования планировщика о завершении текущей работы. Если значение таймера $exec\_timer$ становится равным WCET, то происходит переход в локацию $Send$ для отправки сообщений работам-получателям.

Локация $Send$ соответствует отправке сообщений работам-получателям. В данной локации запрещено продвижение времени. Сразу после перехода в эту локацию, из нее происходит переход в локацию $Finished$ и при этом выполняется отправка сигнала по широковещательному каналу $send$. Этот сигнал получат модели виртуальных каналов, использующихся для передачи сообщений работам-получателям. Если в МВС не имеется задач, синхронно зависящих от данной задачи, то для отправленного сигнала получателей не будет. При этом переход в локацию $Finished$ будет выполнен, т.к. для отправки сигнала по широковещательному каналу не требуется наличие в другом автомате перехода с получением сигнала по этому каналу.

Локация $Finished$ соответствует завершению выполнявшейся на вычислителе работы. В данной локации запрещено продвижение времени. Сразу после перехода в эту локацию, из нее происходит переход в локацию $WaitingForNewPeriod$, во время которого выполняется отправка сигнала по каналу $finished$ и обнуление переменной is_ready.

Локация $WaitingForNewPeriod$ соответствует промежуткам времени между завершением директивного интервала текущей работы и началом периода следующей работы. Эта локация является текущей до тех пор, пока значение $local\_timer$ не достигнет величины периода. После этого выполняется переход в локацию $Dormant$, во время которого обнуляются значения таймеров $local\_timer$ и $exec\_timer$, вызывается функция $resetData()$ для считывания всех пришедших к этому моменту сообщений и значение переменной $abs\_deadline$ увеличивается на значение периода задачи.

В терминах приложения Б описанная модель является периодической с периодом $data.period$, поскольку удовлетворяет достаточным условиям периодичности, сформулированным на стр. 162. Модель работает с четырьмя временными параметрами и имеет один таймер с возможностью останова. В части работы с массивами модель принадлежит классу У6 (см. стр. 165) приложения Б.

Таким образом, достаточные для верификации построенного автомата диапазоны значений параметров определяются согласно утверждения 6 и 11 приложения Б.

А.4. Модель виртуального канала

Параметризованный автомат, моделирующий виртуальный канал, представлен на рисунке А.5.

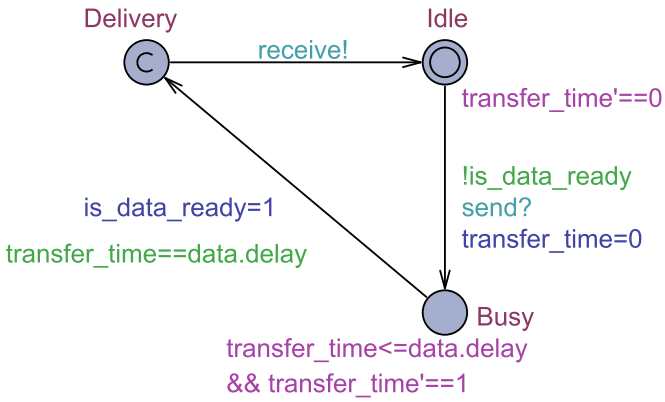

Рисунок А.5 — Модель виртуального канала

Согласно разделу 2.7.4, под виртуальным каналом в данной работе понимается средство передачи сообщений в общем случае. Разработанный автомат моделирует передачу сообщения с фиксированной задержкой.

Согласно разделу 2.4, параметром автомата является задержка на передачу сообщения по данному виртуальному каналу ($data.delay$). Автомат работает с широковещательными каналами $send$ и $receive$, а также с переменной интерфейса is_data_ready. Эта переменная устанавливается моделью виртуального канала в 1, когда очередное сообщение доставлено по виртуальному каналу, и устанавливается в 0 моделью функциональной задачи при считывании полученного сообщения. Также автомат использует таймер $transfer\_time$, активный в локации $Busy$, для измерения времени передачи сообщения.

Начальной локацией автомата является локация $Idle$, соответствующая отсутствию передачи сообщений. Эта локация является текущей до тех пор, пока не будет получен сигнал по каналу $send$. При получении сигнала по каналу $send$ происходит переход в локацию $Busy$. Условием этого перехода является равенство переменной is_data_ready нулю (в противном случае посредством этой переменной работа-получатель может быть ошибочно проинформирована о доставке сообщения до окончания его передачи). При переходе в локацию $Busy$ значение таймера $transfer\_time$ обнуляется.

Локация $Busy$ соответствует передаче сообщения по виртуальному каналу. Эта локация является текущей до тех пор, пока значение таймера $transfer\_time$ не достигнет значения параметра $data.delay$. После этого выполняется переход в локацию $Delivery$ с присвоением переменной is_data_ready значения 1.

Локация $Delivery$ соответствует оповещению работы-получателя о доставке сообщения. В данной локации запрещено продвижение времени. Сразу после перехода в эту локацию, из нее происходит переход в локацию $Idle$ и при этом выполняется отправка сигнала по каналу $receive$.

В терминах приложения Б описанная модель не является периодической, но имеет только один временной параметр $data.delay$, поэтому достаточный для верификации построенного автомата диапазон значений этого параметра определяются согласно утверждению 12 приложения Б.

Приложение Б

Утверждения о выборе диапазонов значений параметров и переменных при верификации

Б.1. Проблема выбора диапазонов значений параметров и переменных при верификации

Согласно разделу 2.7, параметризованные автоматы, моделирующие компоненты МВС, имеют три вида параметров: временные параметры, использующиеся в элементарных сравнениях таймеров (например, длительность передачи сообщения, период задачи, моменты открытия и закрытия окна); параметры, задающие размеры массивов, использующихся в автомате; индексные параметры, использующиеся для доступа к элементам массивов (например, задающие номер раздела, которому принадлежит окно в расписании окон). Также в этих параметризованных автоматах используются переменные.

Разработанные модели (параметризованные автоматы) должны удовлетворять сформулированным в главе 3 требованиям для всех значений своих параметров и при всех последовательностях значений переменных интерфейса, изменяемых вне этих моделей. На практике диапазоны возможных значений параметров слишком велики для того, чтобы при верификации перебирать все возможные их значения. Кроме того, для большинства параметров невозможно априори определить диапазоны их возможных значений. Аналогичная ситуация имеет место для переменных. Для параметризованных временных автоматов известен ряд средств параметрической верификации, позволяющих найти диапазоны значений параметров, при которых выполняется заданное требование. Обзор таких средств приведен в работе [138]. Однако, большая часть описанных в статье средств была создана в рамках исследовательских проектов и в настоящее время не доступна для использования. Доступные же средства не поддерживают все используемые в разработанных моделях конструкции: только IMITATOR [91] поддерживает таймеры с возможностью останова, однако он не поддерживает массивы. Поэтому средства параметрической верификации для проверки выполнения требований к моделям в настоящей работе использоваться не будут, и необходимо свести задачу верификации параметризованных автоматов к задаче верификации автоматов без параметров.

Согласно предложенному подходу, проверка выполнения требований к параметризованному автомату сводится к проверке достижимости «плохой» локации автомата-наблюдателя в сети автоматов, состоящей из исходного автомата и автомата-наблюдателя. В автомате-наблюдателе всегда активны переходы с действиями синхронизации, парными ко всем возможным действиям синхронизации исходного автомата, кроме, быть может, действий синхронизации, не соответствующих корректному функционированию окружения исходного автомата (окружение модели определяется ее интерфейсом, т.е. базовым типом автомата в обобщенной модели МВС,

описанной в разделе 2.4). Если при построении автомата-наблюдателя для проверки некоторого требования A к некоторой модели предполагается, что поведение окружения этой модели корректно, т.е. удовлетворяет некоторому требованию B, то требование A имеет логическую зависимость от требования B. Подробнее про логические зависимости между требованиями см. раздел 3.2.2 стр. 67; в этом разделе на рисунке 3.1 представлен ациклический граф зависимостей между требованиями, определяющий последовательность проверки требований. Также в автомате-наблюдателе выполняется перебор значений параметров и переменных интерфейса исходного автомата. Поскольку модели компонентов МВС должны удовлетворять сформулированным требованиям при условии корректности конфигурации МВС (требования к конфигурации см. в разделе 1.4.1), достаточно перебирать только корректные значения параметров и переменных интерфейса.

Задача верификации сети автоматов (в данном случае проверка достижимости локации) сводится к задаче верификации одного автомата, эквивалентного этой сети. Алгоритм построения такого автомата для сети автоматов приведен в [139]. Для сети автоматов, состоящей из двух автоматов $A_1 = \langle L_1, l_{01}, C_1, V_1, \overline{v_{01}}, AA_1, AS_1, E_1, I_1, P_1, U_1', U_1'' \rangle$ и $A_2 = \langle L_2, l_{02}, C_2, V_2, \overline{v_{02}}, AA_2, AS_2, E_2, I_2, P_2, U_2', U_2'' \rangle$, эквивалентный автомат $A = \langle L, l_0, C, V, \overline{v_0}, AA, AS, E, I, P, U', U'' \rangle$ формируется следующим образом (обозначения см. в разделе 2.2):

- $L = L_1 \times L_2$ — набор локаций;
- $l_0 = (l_{01}, l_{02})$ — начальная локация;
- $C = C_1 \cup C_2$ — набор таймеров;
- $V = V_1 \cup V_2$ — набор переменных;
- $\overline{v_0} = (v_0^1, ..., v_0^n)$, где начальные значения переменных равны соответствующим значениям из $\overline{v_{01}}$ и $\overline{v_{02}}$ (при этом для исходных автоматов должно быть верно, что если некоторая переменная одновременно принадлежит V_1 и V_2, то соответствующие ей значения в $\overline{v_{01}}$ и $\overline{v_{02}}$ должны совпадать);
- $AA = AA_1 \cup AA_2$ — множество действий по обнулению таймеров и изменению переменных;
- $AS = \varnothing$ — действия синхронизации отсутствуют;
- множество размеченных дуг E формируется из элементов перечисленных видов и только их:
  - $(l_{11}, l_{12}) \xmapsto{b_1, \tau, aa_1} (l_{21}, l_{12})$, если $l_{11} \xmapsto{b_1, \tau, aa_1} l_{21} \in E_1$;
  - $(l_{11}, l_{12}) \xmapsto{b_2, \tau, aa_2} (l_{11}, l_{22})$, если $l_{12} \xmapsto{b_2, \tau, aa_2} l_{22} \in E_2$;
  - $(l_{11}, l_{12}) \xmapsto{b_1 \wedge b_2, \tau, (aa_1, aa_2)} (l_{21}, l_{22})$, если $l_{11} \xmapsto{b_1, x!, aa_1} l_{21} \in E_1$ и $l_{12} \xmapsto{b_2, x?, aa_2} l_{22} \in E_2$ (x — канал «точка-точка» или широковещательный канал);
  - $(l_{11}, l_{12}) \xmapsto{b_1 \wedge b_2, \tau, (aa_2, aa_1)} (l_{21}, l_{22})$, если $l_{11} \xmapsto{b_1, x?, aa_1} l_{21} \in E_1$ и $l_{12} \xmapsto{b_2, x!, aa_2} l_{22} \in E_2$ (x — канал «точка-точка» или широковещательный канал);
  - $(l_{11}, l_{12}) \xmapsto{b_1 \wedge \overline{b_2}, \tau, aa_1} (l_{21}, l_{12})$, если $l_{11} \xmapsto{b_1, x!, aa_1} l_{21} \in E_1$ и $l_{12} \xmapsto{b_2, x?, aa_2} l_{22} \in E_2$ (x — широковещательный канал, $\overline{b_2}$ — логическое отрицание выражения b_2);
  - $(l_{11}, l_{12}) \xmapsto{b_1, \tau, aa_1} (l_{21}, l_{12})$, если $l_{11} \xmapsto{b_1, x!, aa_1} l_{21} \in E_1$ и *не существует* таких b_2, aa_2, l_{22}, что $l_{12} \xmapsto{b_2, x?, aa_2} l_{22} \in E_2$ (x — широковещательный канал);
- $I(l_1,l_2) = I_1(l_1) \wedge I_2(l_2)\ \forall (l_1, l_2) \in L$ — инварианты локаций;

– $P((l_1, l_2),c) = P_1(l_1, c) \wedge P_2(l_2, c)\ \forall (l_1, l_2) \in L, c \in C$ — условия активности таймеров;
– $\forall (l_1, l_2) \in L$ верно, что $(l_1, l_2) \in U'$ (U' — множество срочных локаций), если верно хотя бы одно из двух условий: $l_1 \in U'_1$, $l_2 \in U'_2$.
– $\forall (l_1, l_2) \in L$ верно, что $(l_1, l_2) \in U''$ (U'' — множество приоритетных локаций), если верно хотя бы одно из двух условий: $l_1 \in U''_1$, $l_2 \in U''_2$.

В данном приложении сформулированы дополнительные ограничения, накладываемые на автоматы-модели компонентов МВС, а также на автоматы-наблюдатели. Ниже доказано, что если автомат-модель и автомат-наблюдатель удовлетворяет соответствующим ограничениям (выбор комплекта ограничений зависит от устройства автомата — подробнее см. в разделе Б.6), то для проверки выполнения соответствующего автомату-наблюдателю требования к этому автомату-модели на всех допустимых значениях параметров и переменных интерфейса достаточно проверить выполнение этого требования на некотором подмножестве множества допустимых значений параметров и переменных интерфейса. Далее будет обосновано, что для рассматриваемых в данной работе классов автоматов, к которым относятся все разработанные модели, количество элементов в этих подмножествах невелико и позволяет выполнить верификацию за приемлемое на практике время, что было подтверждено для моделей, разработанных автором.

В данном приложении сформулированы и обоснованы ограничения для автоматов с параметрами, используемыми в элементарных сравнениях таймеров, то есть задающими время событий (см. раздел Б.2), параметрами, задающими размеры массивов (см. раздел Б.3), и индексными параметрами (см. раздел Б.4). Другие параметры в данной работе не используются (см. раздел 2.7). Также сформулированы и обоснованы ограничения для автоматов, в которых используются переменные интерфейса (см. раздел Б.5). Правила совместного применения сформулированных ограничений к автоматам приведены в разделе Б.6.

Утверждения, сформулированные в разделах Б.2—Б.5(за исключением утверждений 7 и 8), рассчитаны на применение к автомату, эквивалентному сети из автомата-модели и автомата-наблюдателя и полученному согласно правилам, приведенным на странице 140. В соответствии с этими правилами, в полученном автомате отсутствуют синхронизации. Поэтому вопрос синхронизаций в приведенных ниже утверждениях, за исключением утверждений 7 и 8, не рассматривается.

Б.2. Временные параметры

Временным параметром называется параметр p, используемый только в элементарных сравнениях таймеров, то есть выражениях вида $t\ \ op\ \ p$, где t — таймер, а op — одна из операций $\{<\ ,>\ ,==\ ,!=\ ,>=,<=\}$, а также в конъюнкциях таких выражений. Здесь и далее для обозначения операций используется формат средства UPPAAL, подобный формату языка C. Согласно разделу 2.2, элементарные сравнения вида $(x - y)\ op\ p$ не используются. Согласно разделам 2.1.3 и 2.2, для каждой пары ⟨локация, таймер⟩ задается условие активности таймера в этой локации

— функция от переменных автомата, возвращающая логическое значение. Если для некоторого таймера условия активности во всех локациях тождественно истинны, то такой таймер будем называть не останавливаемым. Под моментами времени в этом разделе понимаются значения неостанавливаемого таймера. Предполагается, что если автомат имеет временные параметры, то неостанавливаемый таймер всегда присутствует в этом автомате. Временные параметры могут принимать только целые неотрицательные значения.

В данной работе считается, что переменные не могут использоваться в операциях сравнения с таймерами. Поскольку, согласно определению используемого математического аппарата (см. раздел 2.2), значения каждой переменной принадлежат конечному множеству целых чисел, для любого автомата, использующего переменные, можно построить автомат без переменных, каждая локация в котором будет соответствовать локации исходного автомата и некоторому набору значений его переменных. При этом количество локаций будет конечным. Таким образом, каждой локации L исходного автомата соответствует набор локаций $\langle L, \overline{v_j} \rangle$, $j \in \overline{1, N_V}$, где $\overline{v_j}$ — j-й набор значений переменных, N_V — количество всевозможных наборов значений переменных. Если в исходном автомате существует переход из локации L_1 в локацию L_2 с условием $cond(\overline{v})$ и действиями $aa(\overline{v})$, то в построенном автомате этому переходу будут соответствовать переходы из $\langle L_1, \overline{v_i} \rangle$ в $\langle L_2, \overline{v_j} \rangle$, для всех таких $\overline{v_i}$ и $\overline{v_j}$, что $cond(\overline{v_i})$ имеет истинное значение, а $\overline{v_j}$ получается из $\overline{v_i}$ при выполнении действий $aa(\overline{v})$. Достижимость некоторой локации L в исходном автомате будет эквивалентна достижимости хотя бы одной из локаций вида $\langle L, \overline{v_j} \rangle$ в построенном автомате. Поэтому утверждения о выборе значений временных параметров при верификации, доказанные для автоматов без переменных, будут справедливы и для автоматов с переменными. Вопрос выбора значений переменных интерфейса при верификации будет рассмотрен далее в разделе Б.5, а вопрос верификации автоматов, имеющих и временные параметры, и переменные интерфейса — в разделе Б.6. В настоящем разделе для краткости будем рассматривать параметризованные автоматы, не имеющие переменных (как внутренних, так и переменных интерфейса).

При верификации не параметризованных временных автоматов используется подход временных регионов и графов регионов [139, 140]. Проиллюстрируем его на простейшем примере. Рассмотрим автомат, представленный на рисунке Б.1. Этот автомат имеет один таймер t.

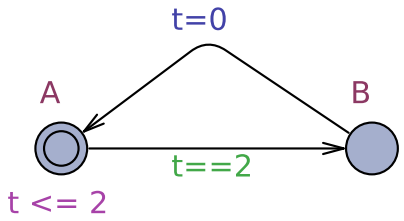

Рисунок Б.1 — Автомат с одним таймером и одной операцией сравнения

Состояние автомата определяется его локацией, значениями переменных и значениями таймеров. В условиях переходов могут использоваться элементарные сравнения таймеров, а также конъюнкции элементарных сравнений. Значения таймеров являются вещественными числами. Таким образом, автомат, представленный на рисунке Б.1, имеет бесконечное число состояний.

Подход временных регионов используется в верификаторах (в т.ч. в UPPAAL) и позволяет объединить состояния, различающиеся лишь значениями таймеров, в одно состояние и перейти

от анализа автомата с бесконечным числом состояний к анализу графа с конечным числом вершин и дуг.

Пусть автомат имеет n таймеров $t_1, ..., t_n$. Два набора значений таймеров $q_1, ..., q_n$ и $q'_1, ..., q'_n$ будем считать *эквивалентными*, если для каждого элементарного сравнения таймеров, используемого в этом автомате, значения сравнения на этих двух наборах совпадают. В соответствии с этим определением, множество всех наборов значений таймеров разбивается на не пересекающиеся временные регионы. Причем такое разбиение не обязательно однозначно и наборы значений таймеров, принадлежащие разным регионам, могут быть эквивалентны.

Неформально, разбиение множества всех наборов значений таймеров на регионы должно удовлетворять следующим условиям [139](*):

1. все наборы значений таймеров, принадлежащие одному региону, эквивалентны;
2. для любого региона $\overline{r}$ однозначно определен регион $\overline{r+}$, следующий за $\overline{r}$ при продвижении времени;
3. для любого фиксированного набора таймеров $t_{i_1}, ..., t_{i_k}$ из общего набора таймеров автомата $t_1, ..., t_n$ верно, что для любого региона $\overline{r}$ обнуление таймеров $t_{i_1}, ..., t_{i_k}$ однозначно определяет регион $\overline{r'}$ (т.е. для любого набора значений таймеров, принадлежащих $\overline{r}$, при обнулении таймеров $t_{i_1}, ..., t_{i_k}$ будет получен набор значений, принадлежащих $\overline{r'}$); $\overline{r'}$ при этом зависит от выбора $i_1, ..., i_k$;
4. количество регионов конечно.

Для автоматов с одним не останавливаемым таймером, либо несколькими не останавливаемыми таймерами, обнуляемыми синхронно (в этом случае значения этих таймеров всегда совпадают и можно считать, что не останавливаемый таймер один) временные регионы можно ввести следующим образом. Пусть имеется автомат с одним не останавливаемым таймером t и различными натуральными константами $m_1, ..., m_k$, а также константой 0, использующимися в элементарных сравнениях с таймером t. Пусть константы упорядочены по возрастанию. Тогда для данного автомата введем следующий набор регионов: $\{0\}$, $(0,m_1)$, $\{m_1\}$, (m_1,m_2), ..., $\{m_k\}$, $(m_k; +\infty)$. Обоснуем, что все требования (*) для введенного таким образом набора регионов выполнены. Регионы для автоматов с одним таймером будем обозначать как r, $r+$, r' и т.п. По построению для всех значений таймера, принадлежащих фиксированному региону, значения элементарного сравнения $t\ op\ m_i$ совпадают (п. 1 в списке (*)). Если $r = \{0\}$, то $r+ = (0,m_1)$; если $r = (m_i, m_{i+1})$, то $r+ = \{m_{i+1}\}$; если $r = \{m_i\}$, то $r+ = (m_i, m_{i+1})$; если $r = \{m_k\}$, то $r+ = (m_k; +\infty)$; если $r = (m_k; +\infty)$, то будем считать, что $r = r+$ (п. 2 в списке (*)). $\{0\}$ — регион, получаемый при обнулении таймера (п. 3 в списке (*)). Число регионов конечно и равно $2k + 2$ (п. 4 в списке (*)).

Подчеркнем, что этот способ определения временных регионов подходит только для случая автомата с одним не останавливаемым таймером, либо несколькими такими таймерами, обнуляемыми синхронно, а также сравнениями вида $t\ op\ m_i$. В противном случае следует использовать более общее определение регионов, приведенное в [141] и предполагающее использование для определения границ регионов в т.ч. *всех* целых чисел от 0 до m_k. Приведенные ниже утверждения 1, 2 и 3 формулируются и доказываются для автоматов этого вида, поэтому вводить общее определение регионов не требуется. К такому виду автоматов относятся разработанные автором модели

планировщика ядра и виртуального канала. Далее будут рассмотрены более сложные частные случаи, предполагающие использование в автомате одного или двух таймеров с возможностью останова, а также от одного до четырех параметров. При формулировании и доказательстве утверждения 4, применимого для этих случаев, аппарат временных регионов не используется. Далее будет показано, что требование синхронности обнуления таймеров существенно для обоснования возможности сокращения перебираемых при верификации значений параметров, и ему должны удовлетворять все автоматы, моделирующие компоненты МВС и имеющие более одного временного параметра. Построенные автором модели удовлетворяют этому требованию.

Пусть $\overline{x}$ — некоторый набор значений таймеров. $B(l)$ — логическое выражение, заданное для некоторой локации l и содержащее конъюнкцию элементарных сравнений этих таймеров. Напомним, что выражение $\overline{x} \in B(l)$ обозначает, что $B(l)$ истинно, если таймеры имеют значения $\overline{x}$. Для разных локаций могут быть заданы разные выражения B. Пусть $\overline{r}$ — некоторый регион. Тогда выражение $\overline{r} \subset B(l)$ будет обозначать, что $\forall \overline{x} \in \overline{r} : x \in B(l)$. По определению региона значение $B(l)$ для всех элементов какого бы то ни было региона одинаково. В т.ч. это верно для инвариантов локаций и условий переходов.

По исходному временному автомату и набору временных регионов для него можно построить граф регионов следующим образом [139, 140]:

Пусть имеется автомат A с набором локаций L, начальной локацией l_0, набором таймеров C, множеством дуг E и назначением инвариантов I (см. обозначения в разделе 2.2). Пусть $R(A)$ — набор регионов для этого автомата, включающий в том числе регион $\{0,...,0\}$, соответствующий нулевым значениям всех таймеров. Тогда граф регионов $RS(A)$ для этого автомата определяется следующим образом:

1. множеством вершин графа регионов является множество $L \times R(A)$;
2. вершина $(l_0, \{0,...,0\})$ называется начальной;
3. в графе присутствует дуга $(l, \overline{r}) \to (l', \overline{r'})$ тогда и только тогда, когда выполняется хотя бы одно из двух условий:
   - $\overline{r'} = \overline{r+}$, $l' = l$, $\overline{r+} \subset I(l)$ (дуга соответствует продвижению времени в исходном автомате, при этом для всех наборов значений таймеров из региона $\overline{r+}$ должен выполняться инвариант текущей локации; поскольку в данном утверждении рассматривается автомат без переменных, то значение инварианта зависит только от значений таймеров);
   - в множестве E имеется переход из l в l' с предусловием b и действиями aa, такой, что:
     - $\overline{r} \subset b$ — предусловие выполнено для всех наборов значений таймеров из $\overline{r}$ (аналогично инварианту локации, при отсутствии переменных предусловие зависит только от значений таймеров);
     - $\overline{r'}$ получается из $\overline{r}$ при выполнении действий из aa по обнулению таймеров (если aa не содержит действий по обнулению таймеров, то $\overline{r'} = \overline{r}$);
     - $\overline{r'} \subset I(l')$ — инвариант локации назначения выполнен для всех наборов значений таймеров из $\overline{r'}$;

Приведенные два случая соответствуют непрерывным и дискретным переходам в исходном автомате.

Для корректного определения графа регионов необходимо, чтобы набор регионов удовлетворял требованиям (*). Так требование 1 необходимо для самой возможности построения дуг по описанным правилам, требования 2 и 3 — для однозначности построения дуг, соответствующих непрерывным и дискретным переходам в исходном автомате, требование 4 — для конечности множества вершин графа регионов.

В случае автомата с одним таймером для определения графа регионов можно использовать набор регионов, сформированный предложенным на странице Б.2 образом на основе значений констант, использующихся в элементарных сравнениях с таймером, так как этот набор регионов удовлетворяет требованиям (*).

Докажем следующее утверждение.

Утверждение 1. Пусть задан автомат A с одним не останавливаемым таймером t и различными натуральными упорядоченными по возрастанию константами $m_1, ..., m_k$ и, возможно, константой 0, использующимися в элементарных сравнениях с таймером t. Пусть $RS(A)$ — граф регионов этого автомата, построенный для набора регионов $\{0\}$, $(0,m_1)$, $\{m_1\}$, (m_1,m_2), ..., $\{m_k\}$, $(m_k; +\infty)$. Тогда для любой локации l' исходного автомата верно, что она достижима из начальной локации тогда и только тогда, когда в графе $RS(A)$ из начальной вершины достижима хотя бы одна из вершин (l',r).

Аналогичное утверждение доказано в [139] для набора регионов, построенных для временных автоматов общего вида, не имеющих переменных, и выполнимости любой формулы темпоральной логики TCTL: «любая формула TCTL выполнима для автомата тогда и только тогда, когда она выполнима для соответствующего графа регионов».

Доказательство.

Адаптируем доказательство из [139] для частного случая автомата с одним таймером, введенного определения регионов, и проверки достижимости локации, т.е. проверки выполнимости формулы TCTL $E <> l'$ (синтаксис формул см. в разделе 2.1.2).

Если (l,c) — некоторое состояние исходного автомата, $[c]$ — регион, которому принадлежит значение таймера c, то $(l,[c])$ — соответствующая этому состоянию вершина в графе $RS(A)$.

Любому вычислению $(l_0,0) \to ... \to (l_i,c^*) \to ...$ в исходном автомате соответствует последовательность вершин $(l_0,\{0\}) \to ... \to (l_j,[c^{**}]) \to ...$ в графе $RS(A)$, достижимых из начальной вершины (по построению $RS(A)$). При этом непрерывному переходу в вычислении исходного автомата может соответствовать одна дуга в $RS(A)$ (переход к следующему региону), цепочка дуг в $RS(A)$ (переход через один или несколько регионов), либо не соответствовать ни одна дуга в $RS(A)$, то есть вершина в $RS(A)$ не меняется (переход в рамках одного региона). Дискретному переходу в вычислении исходного автомата всегда соответствует ровно одна дуга в $RS(A)$.

Сначала докажем, что если локация l' достижима в исходном автомате A, то хотя бы одна из вершин (l',r) достижима в графе $RS(A)$.

Если l' — начальная локация автомата, т.е. $l' = l_0$, то, по построению графа $RS(A)$, его начальной вершиной (а, следовательно, достижимой) вершиной является вершина $(l', \{0,...,0\})$.

Если локация l' не является начальной и достижима в исходном автомате, то в одном из его вычислений существует переход в локацию (l',c'). Рассмотрим последовательно все переходы в этом вычислении $(l_0,0) \to ... \to (l',c')$ и сопоставим им последовательность дуг в графе $RS(A)$. Таким образом получим последовательность дуг в графе $RS(A)$, начинающуюся из начальной вершины $(l_0,\{0\})$ и заканчивающуюся вершиной $(l',[c'])$, что и соответствует достижимости вершины (l',r) ($r = [c']$) в графе $RS(A)$.

Теперь аналогичным образом докажем, что если в графе $RS(A)$ достижима вершина (l',r), то в исходном автомате достижима локация l'.

Если l' — начальная локация автомата, т.е. $l' = l_0$, то она по определению достижима.

Если локация l' не является начальной, то, имея последовательность дуг $(l_0,\{0\} \to ... \to (l', r'))$ в $RS(A)$, можно построить последовательность переходов в исходном автомате, сопоставляя каждой дуге в последовательности дуг $RS(A)$ непрерывный или дискретный переход в исходном автомате в соответствии с определением графа $RS(A)$. Полученная последовательность переходов исходного автомата будет представлять собой начальную часть вычисления этого автомата, на котором достижима локация l'.

Замечание. Из того, что одному непрерывному переходу в вычислении автомата может соответствовать от нуля до нескольких дуг в последовательности дуг $RS(A)$, а при построении вычисления автомата по последовательности дуг в $RS(A)$ каждой дуге соответствует ровно один переход, следует, что если для некоторого вычисления построить соответствующую последовательность дуг, а затем для полученной последовательности дуг построить соответствующее вычисление, то полученное вычисление может не совпадать с начальным.

Утверждение 1 доказано.

Таком образом, доказана эквивалентность достижимости локации в исходном автомате и достижимости хотя бы одной из соответствующих этой локации вершин в графе $RS(A)$.

Рассмотрим теперь, как изменится граф $RS(A)$ при изменении используемых констант.

Пусть имеется автомат A с одним не останавливаемым таймером t и различными натуральными константами $m_1, ..., m_k$ и, возможно, константой 0, использующимися в элементарных сравнениях с таймером t. Пусть константы упорядочены по возрастанию. Пусть автомат A' получается из автомата A заменой констант $m_1, ..., m_k$ на соответственно константы $n_1, ..., n_k$, также различные и упорядоченные по возрастанию. Каждой локации l автомата A соответствует локация l' автомата A', каждой дуге e — дуга e'. Инварианты локаций и условия переходов совпадают с точностью до значений констант.

Введем для автомата A набор регионов $R(A) = (\{0\}, (0,m_1), \{m_1\}, (m_1,m_2), ..., \{m_k\}, (m_k;+\infty))$. Аналогично $R(A') = (\{0\}, (0,n_1), \{n_1\}, (n_1,n_2), ..., \{n_k\}, (n_k;+\infty))$. Каждому региону $r \in R(A)$ взаимно однозначно соответствует регион $r' \in R(A')$: региону $\{0\}$ — регион $\{0\}$, региону $\{m_i\}$ — регион $\{n_i\}$, региону (m_i,m_{i+1}) — регион (n_i,n_{i+1}), региону $(m_k;+\infty)$ — регион $(n_k;+\infty)$.

Каждой вершине графа $RS(A)$ можно взаимно однозначно поставить в соответствие вершину графа $RS(A')$, поскольку между множествами L и L' имеется взаимно однозначное соответствие, а также между множествами $R(A)$ и $R(A')$ имеется взаимно однозначное соответствие.

Для любых $i,j \in \overline{1,k}$ верно, что если значение таймера t принадлежит региону (m_i,m_{i+1}) в автомате A, а значение таймера t' принадлежит региону (n_i,n_{i+1}) в автомате A', то значение элементарного сравнения $t\ op\ m_j$ совпадает со значением $t'\ op\ n_j$. Это следует из того, что оба набора констант упорядочены по возрастанию, и из свойств операций сравнения, используемых в качестве op.

Аналогичный вывод имеет место для регионов вида $\{m_i\}$ и $\{n_i\}$, $(m_k;+\infty)$ и $(n_k;+\infty)$, а также для региона $\{0\}$.

Следовательно, из истинности конъюнкции q элементарных сравнений вида $t\ op\ m_{h_i}$, $i \in \overline{1,q}$ в некотором регионе $r \in R(A)$ следует истинность конъюнкции q элементарных сравнений вида $t'\ op\ n_{h_i}$, $i \in \overline{1,q}$ (с теми же операторами) в соответствующем регионе $r' \in R(A')$. Аналогично для ложности такой конъюнкции.

Докажем, что между дугами графов $RS(A)$ и $RS(A')$ имеется взаимно однозначное соответствие.

Рассмотрим дуги графа $RS(A)$:

1. Пусть имеется дуга $(l,r) \to (l,r+)$. По определению $RS(A)$, $r+ \subset I(l)$; $I(l)$ — инвариант локации l — конъюнкция элементарных сравнений для таймера t. Инвариант соответствующей локации l' в автомате A' представляет собой выражение $I'(l')$, полученное заменой констант m_i на n_i в выражении $I(l)$. Следовательно, $r'+ \subset I'(l')$, а это значит, что в $RS(A')$ имеется дуга $(l,r) \to (l',r'+)$.

2. Пусть имеется дуга $(l_1,r_1) \to (l_2,r_2)$. По определению $RS(A)$, $r_2 \subset I(l_2)$, $r_1 \subset b$, где b (предусловие соответствующего перехода в автомате A) и $I(l_2)$ (инвариант локации l_2) представляют собой конъюнкции элементарных сравнений для таймера t. $I'(l'_2)$ и b' получаются заменой констант m_i на n_i в выражениях $I(l_2)$ и b. Следовательно, $r'_2 \subset I'(l'_2)$, $r'_1 \subset b'$, а это значит, что в $RS(A')$ имеется дуга $(l'_1,r'_1) \to (l'_2,r'_2)$.

Таким образом, доказано, что каждой дуге в графе $RS(A)$ однозначно соответствует дуга в графе $RS(A')$. Аналогичным образом доказывается, что каждой дуге в графе $RS(A')$ однозначно соответствует дуга в графе $RS(A)$. Следовательно, между дугами графов регионов $RS(A)$ и $RS(A')$ имеется взаимно однозначное соответствие, как и между их вершинами. Из этого следует, что некоторая вершина в графе $RS(A)$ достижима тогда и только тогда, когда достижима соответствующая вершина в графе $RS(A')$. В свою очередь из этого, а также из утверждения 1, следует следующее утверждение:

Утверждение 2. Пусть имеется автомат A с одним не останавливаемым таймером t и различными натуральными константами $m_1,...,m_k$ и, возможно, константой 0, использующимися в элементарных сравнениях с таймером t. Пусть константы упорядочены по возрастанию. Пусть автомат A' получается из автомата A заменой констант $m_1,...,m_k$ соответственно на константы $n_1,...,n_k$, также различные и упорядоченные по возрастанию. Каждой локации l автомата A соответствует локация l' автомата A', каждой дуге e — дуга e'. Инварианты локаций и условия переходов совпадают с точностью до замены констант. Тогда для любой локации l автомата A верно, что она достижима тогда и только тогда, когда достижима соответствующая локация l' автомата A'.

Из этого утверждения можно сформулировать следующее следствие:

Пусть имеется автомат A с одним таймером t и различными натуральными константами $m_1, ..., m_k$ и, возможно, константой 0, использующимися в элементарных сравнениях с таймером t. Пусть константы упорядочены по возрастанию. Тогда для любой локации l этого автомата верно, что для того, чтобы проверить достижимость этой локации, достаточно проверить достижимость соответствующей локации в автомате A', полученном из автомата A заменой констант $m_1, ..., m_k$ на константы $1, 2, ..., k$.

Перейдем к параметризованным автоматам.

Утверждение 3. Пусть параметризованный автомат имеет один не останавливаемый таймер t и k временных параметров $p_1, ..., p_k$. Пусть в сравнениях с таймером t могут использоваться только параметры $p_1, ..., p_k$, и, возможно, константа 0. Тогда для любой локации автомата верно, что если она не достижима при любых целых неотрицательных значениях временных параметров, не превышающих k, то она не достижима при любых целых неотрицательных значениях временных параметров.

Доказательство.

Докажем утверждение 3 методом от противного. Предположим, что для некоторого автомата, удовлетворяющего условиям утверждения 3, существует такая локация l, что она не достижима при любых целых неотрицательных значениях временных параметров, не превышающих k, но достижима при некотором наборе целых неотрицательных значений временных параметров, хотя бы одно из которых больше k.

Рассмотрим набор значений временных параметров, при котором локация l достижима. Этот набор представляет собой k целых неотрицательных чисел, некоторые из которых больше k. Исключим из этого набора дубликаты, упорядочим оставшиеся числа в порядке возрастания и пронумеруем. В результате получим набор из $m \leqslant k$ различных целых неотрицательных чисел $a_1, ..., a_m$, упорядоченных по возрастанию. Подставим вместо временных параметров в исходный параметризованный автомат числа из набора $a_1, ..., a_m$ так, чтобы локация l была достижима. Мы получили экземпляр автомата с m (если $a_1 > 0$) или $m - 1$ (если $a_1 = 0$) различными натуральными упорядоченными по возрастанию константами, в котором достижима локация l. При этом a_1 может быть равно 0 как в случае использования в автомате константы 0, так и в случае равенства 0 одного или нескольких временных параметров. Согласно следствию из утверждения 2, локация l достижима и в автомате, полученном заменой констант $a_1, ..., a_m$ на числа $1, ..., m$ (или, если $a_1 = 0$ — заменой $a_2, ..., a_m$ на числа $1, ..., m - 1$). Но, т.к. $m \leqslant k$, это означает, что локация l достижима на некотором наборе значений временных параметров, не превышающих k, что противоречит введенному предположению. Таким образом, введенное предположение неверно, и утверждение 3 доказано.

Далее исследуем, как изменятся интервалы возможных значений временных параметров при добавлении в автомат таймеров с возможностью останова.

Пусть некоторый автомат имеет один таймер t и одну натуральную константу m, использующуюся в элементарных сравнениях с этим таймером. Согласно предыдущим рассуждениям, в этом случае набор регионов можно ввести следующим образом: $\{0\}$, $(0,m)$, $\{m\}$, $(m;+\infty)$.

Теперь рассмотрим автомат, имеющий два таймера $t1$ и $t2$, обнуление которых происходит синхронно. При этом таймер $t2$ может быть остановлен. В элементарных сравнениях таймеров

может по-прежнему использоваться одна натуральная константа m. Из синхронности обнуления таймеров следует, что $t2 \leqslant t1$ в любом состоянии автомата. Для каждого из таймеров в отдельности по-прежнему имеется набор регионов $\{0\}$, $(0,m)$, $\{m\}$, $(m;+\infty)$; для всех значений конкретного таймера, принадлежащих региону, каждое элементарное сравнение *этого* таймера имеет одно и то же значение. Однако, в зависимости от условий запуска/останова таймера $t2$ соотношение значений таймеров может различаться. Пусть таймер $t2$ запущен в момент, когда таймер $t1$ имеет значение m; в таком случае таймер $t2$ примет значение m в момент, когда таймер $t1$ будет иметь значение $2m$. Следовательно, значения элементарного сравнения $t2\ op_2\ m$ могут различаться при $m < t1 < 2m$, $t1 = 2m$ и $t1 > 2m$, в то время как во всех этих случаях значения элементарного сравнения $t1\ op_1\ m$ будут совпадать.

В автомате (см. рисунок Б.2), имеющем три таймера, два из которых могут быть остановлены, выражения вида $t_i == m$ задают три граничные точки $(m, 2m, 3m)$ на оси значений неостанавливаемого таймера $t1$ (в UPPAAL метка локации вида $t' == b$ обозначает, что логическое выражение b — условие активности таймера в этой локации).

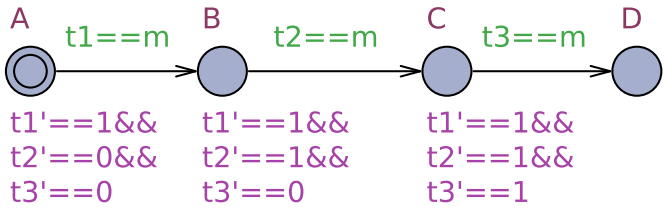

Рисунок Б.2 — Пример автомата с остановкой таймеров

Сформулируем дополнительные ограничения для параметризованных автоматов, содержащих таймеры с возможностью останова, и докажем, что если для автомата с возможностью останова выполнены эти ограничения, то количество перебираемых при верификации значений параметров можно существенно сократить. Для начала рассмотрим автоматы без параметров, содержащие таймеры с возможностью останова. В таких автоматах в сравнениях с таймерами используются только константы. Выводы, полученные для таких автоматов, используются при доказательстве утверждения 4, сформулированного для параметризованных автоматов. В этом доказательстве рассматриваются экземпляры некоторого параметризованного автомата, то есть автоматы, в которых вместо параметров используются константы.

Поскольку в текущих рассуждениях рассматриваются только автоматы без параметров и переменных, то инварианты локаций и условия переходов представляют собой конъюнкции элементарных сравнений таймеров, а условиями активности таймера могут быть только булевы константы 0 и 1. Так как для каждой локации и каждого таймера условие активности таймера в этой локации константно, то для каждого таймера имеется набор локаций, в которых он активен, и набор локаций, в которых он остановлен, а запуск и останов таймеров происходит только при переходах между локациями.

Введем класс Л1 автоматов, не имеющих параметров и удовлетворяющих следующим условиям:

1. Все локации, кроме срочных и непрерывных (в которых запрещено продвижение времени) имеют инварианты, представляющие собой конъюнкции элементарных сравнений

вида $t_k <= m_i$, где t_k — один из таймеров автомата, m_i — некоторая константа; при этом хотя бы один таймер из используемых в инварианте должен быть активным в соответствующей локации.

Это условие означает, что некоторая локация, в которой разрешено продвижение времени, может быть текущей до тех пор, пока значение активного таймера не достигнет значения некоторой константы.

2. Для каждого перехода верно, что либо его условие тождественно истинно, либо его условие представляет собой конъюнкцию элементарных сравнений с операциями $\{<, <=, ==\}$, содержащую хотя бы одно сравнение вида $t_h == m_j$, где t_h — один из таймеров автомата, m_j — одна из констант, и таймер t_h активен в локации-источнике перехода.

   Это условие означает, что либо переход из локации может быть выполнен безусловно, либо переход может быть выполнен строго в момент, когда значение таймера становится равно значению некоторой константы автомата. При этом в общем случае не требуется, чтобы для каждого сравнения вида $t_h == m_j$, использующегося в условии какого-либо перехода из локации, в инварианте этой локации присутствовало сравнение $t_h <= m_j$.
3. В автомате имеется хотя бы один не останавливаемый таймер.

Введенные ограничения соответствуют используемым в данной работе предположениям о том, что длительности всех выполняющихся в МВС операций (передача сообщений, выполнение работ на вычислителе и др.) фиксированы и равны заданным в конфигурации значениям, передаваемым в автоматы в виде констант-значений параметров.

Докажем лемму.

Лемма 1. Пусть в некотором автомате, принадлежащем классу Л1, в элементарных сравнениях таймеров используются k упорядоченных по возрастанию натуральных констант $m_1, ..., m_k$, и, возможно, константа 0. Тогда для любого вычисления S, содержащего переход в некоторую локацию l_i, можно построить другое вычисление S' того же автомата, содержащее переход в локацию l_i, такое что все дискретные переходы в этом вычислении до перехода в локацию l_i включительно будут выполняться в моменты времени, в которые становятся истинными сравнения вида $t_h == m_j$, где t_h — один из таймеров автомата, m_j — одна из констант $m_1, ..., m_k$. Напомним, что моментами времени считаются значения не останавливаемого таймера.

Доказательство.

Будем строить вычисление S' следующим образом. Начальные состояния вычислений S и S' совпадают. Рассмотрим последовательно каждый переход вычисления S, вплоть до перехода в локацию l_i. Переход имеет один из двух видов:

1. Непрерывный переход, то есть продвижение времени до момента времени, которому соответствует некоторый набор значений таймеров $\overline{c}$. Возможно два случая:
   а) Для набора $\overline{c}$ истинно хотя бы одно из сравнений вида $t_h == m_j$, такое, что для него верны два условия:
      1) сравнение $t_h == m_j$ содержится в условии хотя бы одного перехода из текущей локации;
      2) таймер t_h является активным в текущей локации;

Тогда помещаем этот непрерывный переход в вычисление S' в неизменном виде.

б) Для набора $\overline{c}$ не выполняется ни одно из сравнений вида $t_h == m_j$, для которых выполняются условия (1.а.1, 1.а.2). Тогда игнорируем данный непрерывный переход.

Важно, что для вычисления S каждый из дискретных переходов, которые могут быть выполнены непосредственно после рассматриваемого непрерывного перехода вида 1.б, может быть выполнен и из состояния, предшествующего рассматриваемому непрерывному переходу. Это следует из введенных ограничений на вид условий дискретных переходов и инвариантов. Так как для набора $\overline{c}$ не выполняется ни одно из сравнений вида $t_h == m_j$, для которых выполняются условия (1.а.1, 1.а.2), то условия дискретных переходов, возможных после выполнения рассматриваемого непрерывного перехода, могут представлять собой либо булеву константу 1, либо конъюнкцию сравнений вида $t_h == m_j$ (таймер t_h остановлен в текущей локации и имеет значение m_j), $t_h <= m_j$, $t_h < m_j$. Такие условия являются истинными и перед выполнением рассматриваемого непрерывного перехода. Если инвариант локации-назначения, представляющий собой конъюнкцию элементарных сравнений вида $t_h <= m_j$, истинен для набора значений таймеров $\overline{c}$, то он является истинным и для набора значений таймеров, предшествующих выполнению рассматриваемого непрерывного перехода, так как при выполнении такого перехода значения таймеров могут только увеличиваться.

Также если для вычисления S после выполнения рассматриваемого непрерывного перехода может быть выполнен еще один непрерывный переход, продвигающий время до момента, которому соответствует набор значений таймеров $\overline{c'}$, то такой непрерывный переход мог быть выполнен и вместо выполнения рассматриваемого перехода (исходя из ограничений на виды инвариантов).

Таким образом, следующий переход в S также возможен и в S', несмотря на то, что рассматриваемый непрерывный переход не был помещен в S'.

2. Дискретный переход, то есть изменение локации. Помещаем этот переход в вычисление S'. Докажем, что этот переход возможен из текущего состояния S'. Если либо текущее состояние в S и S' является начальным, либо предыдущий переход в S и S' был непрерывным, соответствующим случаю 1.а, то текущие состояния в вычислениях S и S' совпадают, и рассматриваемый переход возможен. Если предыдущий переход был непрерывным, соответствующим случаю 1.б, то согласно рассуждениям п. 1.б, для текущего состояния S' рассматриваемый дискретный переход из S также возможен и в S'. Если предыдущий переход в S и S' был дискретным, то либо состояния в S и S' совпадают (следовательно, рассматриваемый переход возможен в S'), либо рассматриваемому переходу в S непосредственно предшествовало несколько (возможно, один) переходов вида 1.б.

В последнем случае, значения каждого используемого в автомате элементарного сравнения совпадают в вычислениях S и S', согласно рассуждениям п. 1.б и тому, что если в результате выполнения дискретного перехода в S некоторые таймеры обнулились, то и в S' в результате выполнения этого перехода обнулились те же самые таймеры. Поэтому рассматриваемый переход также возможен и в S'.

Таким образом будем строить вычисление S' по вычислению S. Состояния, непосредственно следующие за одним и тем же переходом вида 1.а, будут совпадать в вычислениях S' и S. В остальных случаях соответствующие друг другу состояния этих двух вычислений могут различаться значениями таймеров, но при этом значения каждого используемого в автомате элементарного сравнения в этих состояниях будут совпадать (т.к. в элементарных сравнениях могут использоваться только операторы $<$, $<=$, $==$).

По описанной схеме получаем вычисление S', содержащее переход в локацию l_i, при этом все дискретные переходы в вычислении S' выполняются в моменты времени, в которые становится истинным хотя бы одно из сравнений вида $t_h == m_j$, где t_h — один из таймеров автомата, m_j — одна из констант $m_1, ..., m_k$, что и требовалось доказать.

Из доказанного следует, что если для автомата достижима некоторая локация, то она достижима хотя бы для одного вычисления вида S'. Если локация недостижима ни для одного вычисления вида S', то она не достижима для данного автомата.

Таким образом, для проверки достижимости локации достаточно проверить ее достижимость на всех вычислениях вида S'. Обозначим этот вывод (***). Говоря неформально, вычисление S' получено из вычисления S сдвигом всех дискретных переходов, имеющих место, когда не выполняется ни одно из сравнений вида $t_h == m_j$, удовлетворяющих условиям 1.а.1, 1.а.2, влево по оси значений не останавливаемого таймера до ближайшего момента, в который становится истинным хотя бы одно из сравнений вида $t_h == m_j$, удовлетворяющих условиям 1.а.1, 1.а.2. Это те дискретные переходы, условия которых либо тождественно истинны, либо представляют собой конъюнкцию сравнений, имеющих истинное значение и содержащих только остановленные в текущей локации таймеры.

Далее в этом разделе вводится требование периодичности автоматов: обнуление всех таймеров должно происходить каждый раз по истечении заданного периода, отсчитываемого не останавливаемым таймером, и только в такие моменты времени. Там же приводятся достаточные условия периодичности. В приведенном далее утверждении 4 и предшествующих ему рассуждениях строгая периодичность не требуется, однако требуется, чтобы значение не останавливаемого таймера автомата никогда не превышало значения одной из используемых в этом автомате констант.

Для автомата A с одним не останавливаемым таймером t_1, натуральными упорядоченными по возрастанию константами $m_1, ..., m_k$ и, возможно, константой 0 переходы вычислений вида S' могут выполняться в моменты, когда значения таймера t_1 равны $0, m_1, ..., m_k$. Заменим в автомате A константы $m_1, ..., m_k$ на соответственно числа $1, ..., k$ и получим таким образом автомат A_*. Каждому вычислению вида S' автомата A можно поставить в соответствие вычисление S'_* автомата A_* так, что эти вычисления будут совпадать с точностью до значений таймеров при выполнении переходов: если в вычислении S' некоторый дискретный переход выполняется при $t_1 == m_i$, то в

вычислении S'_* соответствующий переход выполняется при $t_1 == i$; если в вычислении S' при выполнении некоторого непрерывного перехода значение таймера t_1 увеличивается до значения m_i, то в вычислении S'_* при выполнении соответствующего перехода значение таймера t_1 увеличивается до значения i. Важно, что если в некотором состоянии вычисления S' истинно элементарное сравнение $t_1 == m_i$, то в соответствующем состоянии вычисления S'_* истинно элементарное сравнение $t_1 == i$, за счет того, что с увеличением значения t элементарные сравнения вида $t_1 == m_i$ становятся истинными строго в порядке следования значений констант m_i. Таким образом, для любой локации l автомата A верно, что если она достижима для автомата A, то соответствующая локация достижима для автомата A_*. Аналогичным образом (заменой 1, ..., k на m_1, ..., m_k) показывается, что из достижимости некоторой локации l для автомата A_* следует достижимость соответствующей локации для автомата A. Этот вывод совпадает с выводом утверждения 2. Продолжим данные рассуждения для случая автоматов с останавливаемыми таймерами и тем самым перейдем к более общему случаю.

Добавление в автомат таймера t_2 с возможностью останова увеличивает количество моментов, в которые могут выполняться переходы: так, например, если таймер t_2 был активирован в момент $t_1 == m_i$ и имел в этот момент значение 0, то элементарное сравнение $t_2 == m_j$ станет истинным в момент $t_1 == m_i + m_j$, если в течении m_j единиц времени с момента $t_1 == m_i$ таймер t_2 не останавливался.

Количество моментов времени на оси значений не останавливаемого таймера, в которые один из таймеров достигает значения одной из констант, используемых в автомате, зависит от количества констант и таймеров в автомате, а также от условий активности таймеров. Найдем такие моменты времени для нескольких случаев, по совокупности покрывающих все автоматы, используемые в данной работе. Во всех случаях предполагается, что для автоматов выполняются все введенные ранее ограничения, а именно:

- автоматы принадлежат классу Л1, введенному на странице 149;
- обнуление таймеров происходит синхронно;
- в элементарных сравнениях используются только натуральные константы и, возможно, константа 0.

Также, в отличие от рассуждений выше, где значения используемых в автоматах констант были различны и упорядочены по возрастанию, в последующих рассуждениях предполагается, что если соотношение значений используемых в автоматах констант не введено явно, то это соотношение может быть произвольным (в т.ч. значения могут совпадать).

Поскольку условия активности таймеров в локациях являются логическими константами, то запуск и останов таймеров происходит при выполнении дискретных переходов. Обнуление таймеров также происходит при выполнении дискретных переходов. Согласно выводу (***) со стр. 152, следующему из леммы 1, для проверки (не)достижимости некоторой локации достаточно рассматривать дискретные переходы, выполняемые в моменты времени, в которые становится истинным хотя бы одно из сравнений вида $t_h == m_j$, где t_h — один из таймеров автомата, m_j — одна из констант, используемых в сравнениях с таймерами.

Рассмотрим следующие случаи:

а) Автомат с двумя таймерами t_1 и t_2. Таймер t_1 — не останавливаемый таймер, для которого в элементарных сравнениях может использоваться константа m_1. Значение таймера t_1 не превышает константы m_1. Таймер t_2 может быть остановлен. Для него в элементарных сравнениях может использоваться константа m_2.

Условие $t_1 == m_1$ является истинным в момент m_1, поскольку моменты времени введены как значения не останавливаемого таймера.

Таймер t_2 может быть запущен в моменты $0, m_1, m_2$ (запуск по условию $t_2 == m_2$ мгновенно после останова по условию $t_2 == m_2$). Однако, если момент m_1 достигается, то в момент m_1 оба таймера должны быть обнулены, поэтому запуск в момент m_1 неотличим от запуска в момент 0.

Если $m_2 > m_1$, то $t_2 == m_2$ всегда ложно. Если $m_2 \leqslant m_1$, то $t_2 == m_2$ может стать истинным в момент m_2.

Итого, достаточно рассматривать дискретные переходы, выполняемые в моменты $0, m_1, m_2$.

б) Автомат с двумя таймерами t_1 и t_2. Таймер t_1 — не останавливаемый таймер, для которого в элементарных сравнениях могут использоваться константы m_{11}, m_{12}, такие что $m_{11} < m_{12}$. Значение таймера t_1 не превышает константы m_{12}. Таймер t_2 может быть остановлен. Для него в элементарных сравнениях может использоваться константа m_2, такая что $m_2 \leqslant m_{12}$.

Условия $t_1 == m_{11}$, $t_1 == m_{12}$ истинны в моменты m_{11}, m_{12} соответственно.

Условие $t_2 == m_2$ может стать истинным в моменты m_2 (при запуске t_2 в момент 0), $m_{11} + m_2$ (при запуске t_2 в момент m_{11}; при этом должно выполняться неравенство $m_{11} + m_2 < m_{12}$).

Итого, достаточно рассматривать дискретные переходы, выполняемые в моменты 0, m_{11}, m_{12}, m_2, а также в момент $m_{11} + m_2$ (рассматриваются при $m_{11} + m_2 < m_{12}$).

в) Автомат с двумя таймерами t_1 и t_2. Таймер t_1 — не останавливаемый таймер, для которого в элементарных сравнениях могут использоваться константы m_{11}, m_{12}, m_{13}, такие что $m_{11} < m_{12} < m_{13}$. Значение таймера t_1 не превышает константы m_{13}. Таймер t_2 может быть остановлен. Для него в элементарных сравнениях может использоваться константа m_2, такая что $m_2 \leqslant m_{13}$.

Условия $t_1 == m_{11}$, $t_1 == m_{12}$, $t_1 == m_{13}$ истинны в моменты m_{11}, m_{12}, m_{13} соответственно.

Условие $t_2 == m_2$ может стать истинным в моменты m_2 (при запуске t_2 в момент 0), $m_{11} + m_2$ (при запуске t_2 в момент m_{11}; при этом должно выполняться неравенство $m_{11} + m_2 < m_{13}$), $m_{12} + m_2$ (при запуске t_2 в момент m_{12}; при этом должно выполняться неравенство $m_{12} + m_2 < m_{13}$), $m_{12} + (m_2 - m_{11})$ (при запуске t_2 в момент 0, останове в момент m_{11} и следующем запуске в m_{12}; при этом должны выполняться неравенства $m_{11} < m_2$, $m_{12} + (m_2 - m_{11}) < m_{13}$). В выражении $m_{12} + (m_2 - m_{11})$ разность $(m_2 - m_{11})$ — это разница между значением таймера t_2 в момент останова (момент m_{11}) и константой, с которой этот таймер сравнивается (m_2). Выражение $m_{12} + (m_2 - m_{11})$ можно переписать как $m_2 + (m_{12} - m_{11})$, где $(m_{12} - m_{11})$ — время простоя таймера t_2. Других моментов, в которые элементарное сравнение $t_2 == m_2$ может стать истинным, нет.

Итого, достаточно рассматривать дискретные переходы, выполняемые в моменты 0, m_{11}, m_{12}, m_{13}, m_2, а также в моменты $m_{11} + m_2$ (рассматриваются при $m_{11} + m_2 < m_{13}$), $m_{12} + m_2$

(рассматриваются при $m_{12} + m_2 < m_{13}$), $m_{12} + (m_2 - m_{11})$ (рассматриваются при $m_{11} < m_2$, $m_{12} + (m_2 - m_{11}) < m_{13}$).

г) Добавим в автомат, рассмотренный в п. в), еще один таймер t_2^* с возможностью останова, для которого, как и для таймера t_2, в элементарных сравнениях может использоваться константа m_2. Этот случай соответствует наиболее сложной из использующихся в работе композиций автомата-модели и автомата-наблюдателя, а именно модели функциональной задачи и автомата-наблюдателя, проверяющего, что очередная выполняющаяся работа задачи завершается, как только время ее выполнения на процессоре становится равным WCET.

Элементарные сравнения $t_2 == m_2$, $t_2^* == m_2$ в такой сети автоматов могут стать истинными во все соответствующие моменты, перечисленные в п. в), а также в моменты $m_2 + m_2$ (при $m_2 + m_2 < m_{13}$), $m_{11} + m_2 + m_2$ (при $m_{11} + m_2 + m_2 < m_{13}$), $m_{12} + m_2 + m_2$ (при $m_{12} + m_2 + m_2 < m_{13}$), $m_{12} + (m_2 - m_{11}) + m_2$ (при $m_{11} < m_2$, $m_{12} + (m_2 - m_{11}) + m_2 < m_{13}$). Дополнительные по сравнению с п. в) моменты времени возникают из-за возможности запуска останавливаемых таймеров «цепочкой»: запуск одного в момент, когда значение другого достигает m_2.

Итого, для наиболее сложного из рассматриваемых случаев (г) достаточно рассматривать дискретные переходы, выполняемые в следующие моменты: 0, m_{11}, m_{12}, m_{13}, m_2, $m_{11} + m_2$, $m_{12} + m_2$, $m_{12} + (m_2 - m_{11})$, $m_2 + m_2$, $m_{11} + m_2 + m_2$, $m_{12} + m_2 + m_2$, $m_{12} + (m_2 - m_{11}) + m_2$ — всего 12 точек на оси значений не останавливаемого таймера. При этом не для любых значений констант m_{11}, m_{12}, m_{13}, m_2 переходы возможны во все перечисленные моменты: например, дискретный переход не может произойти в момент $m_{12} + (m_2 - m_{11})$, если выполняется хотя бы одно из неравенств $m_{12} + (m_2 - m_{11}) > m_{13}$, $m_{11} > m_2$. Если $m_{12} + (m_2 - m_{11}) = m_{13}$ или $m_{11} = m_2$, то дискретный переход в момент $m_{12} + (m_2 - m_{11})$ возможен, но этот момент совпадает с одним из других перечисленных моментов. Перечисленные моменты являются лишь *возможными* моментами дискретных переходов, и разным итерациям функционирования автомата могут соответствовать различные моменты времени из перечисленных моментов.

В зависимости от значений m_{11}, m_{12}, m_{13}, m_2 перечисленные 12 моментов времени могут быть упорядочены по-разному. Следовательно, элементарные сравнения таймеров могут становиться истинными в разном порядке. Поэтому, в отличие от автоматов с одним не останавливаемым таймером, при замене в некотором автомате A упорядоченных констант m_{11}, m_{12}, m_{13}, m_2 на константы 1, 2, 3, 4 будет получен автомат A_* такой, что из (не)достижимости локации l в этом автомате вообще говоря НЕ следует (не)достижимость локации в исходном автомате A.

С раскрытыми скобками перечень моментов времени принимает вид: 0, m_{11}, m_{12}, m_{13}, m_2, $m_{11} + m_2$, $m_{12} + m_2$, $m_{12} + m_2 - m_{11}$, $2m_2$, $m_{11} + 2m_2$, $m_{12} + 2m_2$, $m_{12} + 2m_2 - m_{11}$. Обозначим выражения 0, x_1, x_2, x_3, x_4, $x_2 + x_1$, $x_3 + x_1$, $x_3 + x_1 - x_2$, $2x_1$, $x_2 + 2x_1$, $x_3 + 2x_1$, $x_3 + 2x_1 - x_2$, задающие функции от переменных x_1, x_2, x_3, x_4, как $f_1(x_1,x_2,x_3,x_4), ..., f_{12}(x_1,x_2,x_3,x_4)$ соответственно. Построим для констант m_{11}, m_{12}, m_{13}, m_2, таких что $m_{11} < m_{12} < m_{13}$, $m_2 \leqslant m_{13}$ (см. п. г)), набор значений $a_1, ..., a_k$, $k \leqslant 12$ следующим образом: вычислим значения функций $f_1(m_{11}, m_{12}, m_{13}, m_2), ..., f_{12}(m_{11}, m_{12}, m_{13}, m_2)$, удалим из полученного набора дубликаты и упорядочим оставшиеся значения по возрастанию. Теперь рассмотрим некоторый набор констант n_{11},

n_{12}, n_{13}, n_2 и аналогичным образом построим для них набор значений $a^*_1, ..., a^*_{k^*}$. Предположим, что для констант n_{11}, n_{12}, n_{13}, n_2 выполнены следующие условия ♠:

– $k = k^*$;
– для любых $i \leqslant k, j \leqslant 12$, таких что $a_i = f_j(m_{11}, m_{12}, m_{13}, m_2)$, верно, что $a^*_i = f_j(n_{11}, n_{12}, n_{13}, n_2)$

Константы n_{11}, n_{12}, n_{13}, n_2, удовлетворяющие условиям ♠, всегда существуют: в частности, этим условиям удовлетворяют $n_{11} = m_{11}$, $n_{12} = m_{12}$, $n_{13} = m_{13}$, $n_2 = m_2$. Схема получения констант n_{11}, n_{12}, n_{13}, n_2, удовлетворяющих условиям ♠ и имеющих небольшие целочисленные значения, будет описана ниже. Согласно приведенным далее рассуждениям, чем меньше значения этих констант, тем меньшее количество значений параметров нужно перебирать при верификации автомата.

Рассмотрим автомат A описанного в п. г) вида. Построим автомат A_*, заменив в автомате A константы m_{11}, m_{12}, m_{13}, m_2 на константы n_{11}, n_{12}, n_{13}, n_2, удовлетворяющие описанным выше условиям ♠ (предположим, что их удалось найти). Автомат A удовлетворяет условиям леммы 1. Выполним для этого автомата рассуждения, аналогичные рассуждениям, приведенным на стр. 152 для случая автомата без таймеров с возможностью останова, и отличающиеся лишь тем, что вместо замены констант m_1, ..., m_k на константы 1, ..., k выполняется замена констант m_{11}, m_{12}, m_{13}, m_2 на константы n_{11}, n_{12}, n_{13}, n_2. В результате этих рассуждений получим, что каждому вычислению вида S' автомата A можно поставить в соответствие вычисление S'_* автомата A_* так, что эти вычисления будут совпадать с точностью до значений таймеров при выполнении переходов. Следовательно, для любой локации l автомата A верно, что она достижима тогда и только тогда, когда соответствующая локация достижима для автомата A_*.

Опишем способ получения констант n_{11}, n_{12}, n_{13}, n_2, удовлетворяющих условиям ♠ и имеющих небольшие целочисленные значения. Приведенным выше (в п. в)) ограничениям на значения констант ($m_{11} < m_{12} < m_{13}$, $m_2 \leqslant m_{13}$), соответствуют ограничения на значения функций $f_1, ..., f_{12}$: например, $f_2(x_1,x_2,x_3,x_4) < f_3(x_1,x_2,x_3,x_4)$, т.к. $m_{11} < m_{12}$, и т.д. Кроме того, $f_1(x_1,x_2,x_3,x_4) \equiv 0$. Из этих ограничений имеются следствия: например, из $f_2(x_1,x_2,x_3,x_4) < f_3(x_1,x_2,x_3,x_4)$ следует, что $f_6(x_1,x_2,x_3,x_4) < f_7(x_1,x_2,x_3,x_4)$, т.к. $m_{11} + m_2 < m_{12} + m_2$, и т.п. С помощью программы, перебирающей все возможные варианты следования значений функций $f_1, ..., f_{12}$ на числовой оси (с учетом возможных совпадений значений) и отсекающей те из них, которые не удовлетворяют ограничениям, получено 316 вариантов следования значений этих функций. Для каждого варианта был подобран такой набор значений x_1, x_2, x_3, x_4, для которого этот вариант имеет место, и которые можно взять в качестве констант n_{11}, n_{12}, n_{13}, n_2 для этого варианта. При этом максимальное из использованных в таких наборах значений равно 22; оно используется в т.ч. для следующего варианта следования значений функций: f_1, f_2, f_5, f_6, f_3, f_9, f_{10}, f_8, f_7, f_{12}, f_4, f_{11}. Этот вариант имеет место при $x_1 = 2$, $x_2 = 11$, $x_3 = 22$, $x_4 = 6$.

Так как максимальное из значений, использованных в наборах x_1, x_2, x_3, x_4, равно 22, то для любого упорядочения функций $f_1, ..., f_{12}$, удовлетворяющего описанным выше ограничениям-неравенствам, можно подобрать такой набор значений их аргументов x_1, x_2, x_3, x_4, не превышающих 22, что значения функций при этих значениях аргументов будут упорядочены по

возрастанию. При полном переборе целочисленных значений x_1, x_2, x_3, x_4 от 1 до 22 такой набор гарантированно будет одним из рассмотренных.

Для автоматов с меньшим, чем в п. г), количеством констант и таймеров имеют место аналогичные рассуждения с тем лишь отличием, что количество возможных моментов дискретных переходов в вычислениях вида S' будет меньше. Так для случая в) возможные моменты таких переходов определяются восемью функциями, и для любого (в рамках ограничений, соответствующих п. в)) упорядочения этих функций можно подобрать такие значения переменных x_1, x_2, x_3, x_4, не превышающие 10, что значения функций при этих значениях переменных будут упорядочены по возрастанию. Для случаев а) и б) значения переменных, перебираемых для получения всех возможных вариантов следования значений соответствующих функций, можно ограничить еще меньшей константой.

Аналогичные рассуждения можно выполнить и для автоматов с бóльшим количеством таймеров и констант: найти все возможные функции от значений констант, определяющие возможные моменты дискретных переходов в вычислениях вида S', а также неравенства, связывающие эти функции; получить все возможные упорядочения этих функций с учетом выполнения неравенств; для каждого упорядочения найти набор значений констант, при которых оно имеет место, и выделить максимальное из найденных значений x_{max}. Для полученного значения x_{max} будет верно, что для любого удовлетворяющего неравенствам упорядочения найденных функций можно подобрать такие значения аргументов этих функций, не превышающие x_{max}, что значения функций при этих значениях аргументов будут упорядочены по возрастанию. Однако, в данной работе для дальнейших рассуждений достаточно выводов, полученных для частных случаев а)—г).

Имеет место следующее утверждение.

Утверждение 4. Пусть имеется параметризованный автомат, для которого выполняются условия:

1. количество временных параметров не превышает четырех;
2. количество таймеров с возможностью останова не превышает двух;
3. все таймеры с возможностью останова сравниваются только с одним временным параметром; при этом неостанавливаемый таймер с этим параметром сравниваться не может;
4. обнуление всех таймеров происходит синхронно; поэтому можно считать, что имеется один неостанавливаемый таймер (т.к. значения синхронно обнуляемых неостанавливаемых таймеров всегда совпадают);
5. выполняются ограничения определения класса Л1 (см. стр. 149) на вид условий переходов и инвариантов локаций с поправкой на то, что вместо констант используются параметры (т.е. каждый экземпляр рассматриваемого параметризованного автомата принадлежит классу Л1);
6. значения всех таймеров не превышают наибольшего из значений временных параметров автомата, использующихся в сравнениях с неостанавливаемым таймером;
7. условия активности таймеров не содержат параметров.

Тогда из недостижимости некоторой локации этого автомата при любых значениях временных параметров, не превышающих 22, следует недостижимость этой локации при любых значениях временных параметров.

Доказательство.

Докажем утверждение 4 методом от противного. Предположим, что для некоторого автомата, удовлетворяющего условиям утверждения 4, существует такая локация l, что она не достижима при любых значениях параметров, не превышающих 22, но достижима при некотором наборе значений параметров, хотя бы одно из которых больше 22.

Рассмотрим набор значений параметров, при котором локация l достижима (т.е. набор констант).

Сначала рассмотрим случай, когда временных параметров четыре, все значения параметров не равны 0, и значения параметров, используемых в сравнениях с неостанавливаемыми таймерами, различны. Константу, использующуюся в сравнениях с таймерами с возможностью останова, обозначим как m_2. Остальные константы обозначим в порядке возрастания как m_{11}, m_{12}, m_{13}. Таким образом, будут выполняться неравенства $m_{11} < m_{12} < m_{13}$. Рассмотрим некоторый экземпляр A автомата, в котором вместо временных параметров используются эти константы и в котором достижима локация l. A можно рассматривать как автомат без параметров.

Так как по условию 6 значения всех таймеров не превышают наибольшего из значений временных параметров автомата, использующихся в сравнениях с неостанавливаемым таймером, то значение таймера с возможностью останова не превышает m_{13}. Рассмотрим два варианта: $m_2 > m_{13}$ и $m_2 \leqslant m_{13}$.

Если $m_2 > m_{13}$, то в силу условия 6 каждое используемое в автомате сравнение вида $t\ op\ m_2$ имеет константное логическое значение, независимо от состояния автомата. Заменив в автомате A все сравнения вида $t\ op\ m_2$ на соответствующие логические константы, получим автомат A', в котором достижима локация l, не используются (то есть не фигурируют ни в каких сравнениях) таймеры с возможностью останова, а в сравнениях с неостанавливаемым таймером используются константы $m_{11} < m_{12} < m_{13}$. Тогда, согласно следствию из утверждения 2, в автомате A'', полученном из автомата A' заменой m_{11}, m_{12}, m_{13} на числа 1, 2, 3, локация l достижима. Теперь заменим в автомате A'' логические константы, полученные при построении автомата A' на основе автомата A, на сравнения вида $t\ op\ 4$. Это всегда возможно, т.к., согласно рассуждениям выше, значения всех таймеров не превышают 3. Таким образом получим автомат A''', в котором достижима локация l и который представляет собой экземпляр исходного параметризованного автомата со значениями параметров, равными 1, 2, 3 и 4, где 4 — значение временного параметра, используемого в сравнениях с таймерами с возможностью останова. Следовательно, локация l исходного параметризованного автомата достижима для найденного набора значений параметров, каждое из которых не превышает 4, что противоречит введенному предположению. Таким образом, введенное предположение неверно.

Если $m_2 \leqslant m_{13}$, то выполняются все неравенства, приведенные выше в п.п. в) и г), и, согласно рассуждениям выше, этому набору констант можно сопоставить набор натуральных констант n_{11}, n_{12}, n_{13}, n_2, не превышающих 22, таких что в автомате, полученном из исходного автомата заменой m_{11}, m_{12}, m_{13}, m_2 на n_{11}, n_{12}, n_{13}, n_2, локация l также достижима, что противоречит введенному предположению. Следовательно, введенное предположение неверно.

Теперь рассмотрим случай, когда некоторые из значений параметров, сравниваемых с неостанавливаемыми таймерами, совпадают, либо когда некоторые из значений параметров равны 0. Этот набор значений задает автомат, в котором используется *менее четырех* натуральных констант.

Аналогично рассуждениям для первого случая, можно показать, что либо все сравнения с останавливаемым таймером имеют константное логическое значение независимо от состояния автомата (тогда может быть применено следствие из утверждения 2), либо константа, использующаяся в сравнениях с останавливаемым таймером, не превышает максимального из значений констант, использующихся в сравнениях с неостанавливаемыми таймерами (тогда имеет место случай, описанный в одном из п. а) и б), для которого выполняются рассуждения, аналогичные рассуждению для п. г)). В обоих вариантах существует такой набор констант, не превышающих 22 (для первой альтернативы — не превышающих 3, для второй — не превышающих 10), что при замене исходных констант на эти константы получается автомат, в котором, также как и в исходном автомате, достижима локация l, что противоречит введенному предположению. Следовательно, введенное предположение неверно.

Таким образом, рассмотрены все возможные количества различных временных параметров автомата, и для каждого из случаев обосновано, что введенное предположение неверно. Утверждение 4 доказано.

Замечание. Условия утверждения 4 соответствуют наиболее сложной из используемых в работе композиций автомата-модели компонента МВС и автомата-наблюдателя. Не рекомендуется создавать модели, выходящие по количеству таймеров с возможностью останова за рамки ограничений утверждения 4, то есть такие, что в сети автоматов, состоящей из автомата-модели и автомата-наблюдателя, имеется более двух таймеров с возможностью останова, так как это *может* привести к алгоритмической неразрешимости задачи верификации [95]. На выбранном уровне абстракции моделирования таймеры с возможностью останова используются только для моделирования вытеснения работ — в модели функциональной задачи достаточно одного такого таймера. А для верификации модели функциональной задачи достаточного одного таймера с возможностью останова в автомате-наблюдателе. Таким образом, суммарное количество таймеров с возможностью останова в сети автоматов, используемой при верификации, не превышает двух. Если существует потребность в таких моделях, что сеть автоматов, необходимая для их верификации, будет выходить по количеству таймеров с возможностью останова за рамки ограничений утверждения 4, то рекомендуется представлять такие модели в виде композиции нескольких более простых автоматов, каждый из которых верифицируется отдельно. Например, в данном диссертационном исследовании для моделирования выполнения работ в рамках раздела используется не одна модель планировщика работ раздела, хранящая информацию о временных характеристиках всех работ (такая модель потребовала бы столько таймеров с возможностью останова, сколько задач имеется в разделе), а модель планировщика работ раздела, не имеющая таймеров и временных параметров и взаимодействующая с моделями всех функциональных задач раздела (модель одной задачи содержит один таймер с возможностью останова). Верификация модели планировщика работ раздела и модели функциональной задачи выполняется раздельно.

Из выполнения условий утверждения 3 для каждого из двух автоматов сети в общем случае не следует их выполнение для автомата, эквивалентного этой сети. Далее будут введены дополнительные ограничения на каждый из двух автоматов сети, из выполнения которых, а также из выполнения условий утверждения 3 для каждого из автоматов сети, будет следовать выполнение условий утверждения 3 для автомата, эквивалентного этой сети. Аналогично для утверждения 4.

Рассмотрим пример, представленный на рисунке Б.3. Сеть автоматов состоит из двух автоматов $A1$ и $A2$ с общими параметрами $p1$ и $p2$. Автомат $A1$ имеет неостанавливаемый таймер $t1$, автомат $A2$ — неостанавливаемый таймер $t2$. Каждый автомат в отдельности удовлетворяет условиям утверждения 3: параметры используются только в элементарных сравнениях таймеров, таймеры каждого автомата обнуляются синхронно. Однако в сети автоматов в целом обнуление таймеров не происходит синхронно. В момент отправки сигнала по каналу s таймер $t1$ имеет значение $p1$, а таймер $t2$ — значение $7 \cdot p1$. Таким образом, локация $E2$ достижима, когда $p2 > 7 \cdot p1$, и из того, что локация $E2$ не достижима при положительных значениях параметров $p1$ и $p2$, не превышающих 2, не следует недостижимость этой локации при всех возможных значениях параметров.

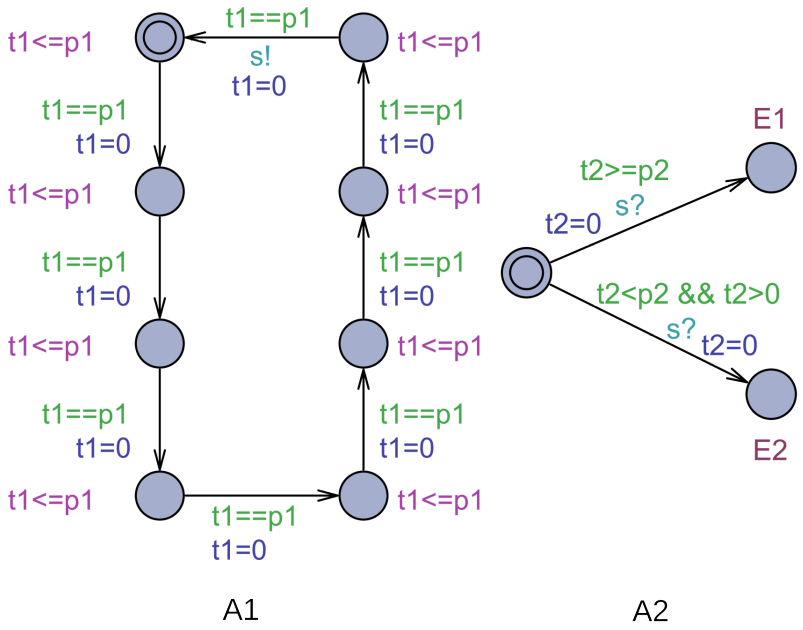

Рисунок Б.3 — Пример сети автоматов с двумя таймерами и двумя параметрами

Пример демонстрирует, что если таймеры сети автоматов обнуляются не синхронно, то величина, на которую их значения могут разойтись, в общем случае не известна. В то же время соотношение величины этого расхождения и значений параметров может влиять на достижимость некоторой локации.

Рассмотрим условия утверждения 3 в применении к автомату, эквивалентному сети автоматов из автомата-модели и автомата-наблюдателя, с учетом правил построения такого автомата (см. стр. 140). Ограничения на вид элементарных сравнений выполняются для этого автомата тогда и только тогда, когда они выполняются для каждого из двух автоматов сети в отдельности. Количество временных параметров для этого автомата совпадает с количеством временных параметров

для автомата-модели, поскольку по определению автомат-наблюдатель не имеет собственных параметров и работает с параметрами автомата-модели. Таким образом, из выполнения всех условий утверждения 3 для каждого из двух автоматов рассматриваемой сети следует выполнение всех условий этого утверждения, кроме условия синхронности обнуления таймеров, для автомата, эквивалентного этой сети автоматов.

Теперь рассмотрим условия утверждения 4 в применении к автомату, эквивалентному сети автоматов из автомата-модели и автомата-наблюдателя, с учетом правил построения такого автомата. Условие 1 выполняется тогда и только тогда, когда количество временных параметров автомата-модели не превышает четырех (аналогично рассуждениям для утверждения 3). Условие 2 — тогда и только тогда, когда *суммарное по автомату-модели и автомату-наблюдателю* количество таймеров с возможностью останова не превышает двух. Условие 3 — тогда и только тогда, когда это условие выполняется для каждого из двух рассматриваемых автоматов в отдельности и, кроме того, временной параметр, с которым сравниваются значения таймеров с возможностью останова, — один и тот же для обоих автоматов. Подход к обоснованию выполненности условия 4 (синхронность обнуления таймеров) будет приведен ниже. Условия 5 и 6 — тогда и только тогда, тогда они выполняются для каждого из двух рассматриваемых автоматов в отдельности.

Таким образом для утверждений 3 и 4 остается либо сформулировать дополнительные ограничения на автомат-модель и автомат-наблюдатель и обосновать, что из выполнения этих ограничений следует синхронное обнуление таймеров в сети из этих двух автоматов (это сделано для периодических автоматов), либо выделить случаи, когда для применения утверждений 3 и 4 условие синхронности обнуления таймеров можно снять (это сделано для непериодических автоматов).

Ниже приведены рассуждения для утверждения 4. Рассуждения для утверждения 3 аналогичны рассуждениям для утверждения 4, с тем лишь отличием, что останавливаемых таймеров в рассматриваемых автоматах нет, а количество временных параметров может быть произвольным.

а) Периодические автоматы.

В таких автоматах имеется неостанавливаемый таймер, который обнуляется строго один раз в конце заданного с помощью параметра периода. Помимо этого таймера в автомате могут использоваться и другие таймеры, останавливаемые в некоторых локациях, и обнуляемые синхронно с первым (неостанавливаемым) таймером. Для верификации таких автоматов используются автоматы-наблюдатели, как правило функционирующие также периодически с тем же самым периодом. Вообще говоря, автоматы-наблюдатели могут и не быть периодическими, но в данной работе все разработанные для периодических автоматов-моделей автоматы-наблюдатели являются периодическими и имеют период, совпадающий с периодом соответствующих моделей. В этом случае таймеры исходного автомата и автомата-наблюдателя обнуляются синхронно и, следовательно, для автомата, эквивалентного рассматриваемой сети автоматов, выполняется условие 4 утверждения 4.

Для того, чтобы параметризованный автомат, имеющий неостанавливаемый таймер t, был периодическим с периодом p (p — временной параметр), достаточно выполнения следующих условий:

1. Для каждого перехода, в действиях которого имеется действие $t = 0$, верно, что условие этого перехода либо имеет вид $t == p$, либо представляет собой конъюнкцию выражения $t == p$ и произвольного логического выражения.
2. Для каждой локации, в которой разрешено продвижение времени, верно, что инвариант этой локации либо имеет вид $t <= p_i$ (или $t < p_i$), либо представляет собой конъюнкцию выражения $t <= p_i$ (или $t < p_i$) и произвольного логического выражения (p_i — один из временных параметров автомата; в т.ч. в качестве p_i может выступать p).
3. Значения всех временных параметров не превышают значения параметра p.

Условие 1 означает, что значение неостанавливаемого таймера может быть обнулено только тогда, когда оно равно значению периода. Условия 2 и 3 означают, что значение неостанавливаемого таймера не может превышать значения периода. Таким образом, из выполнения всех трех условий следует периодичность автомата.

Все разработанные автором автоматы-наблюдатели и автоматы-модели являются периодическими.

После проверки выполнения условий 1—3 и 5—6 для автомата-модели и автомата-наблюдателя, а также проверки выполнения условий периодичности автомата-модели сеть автоматов, состоящую из исходного автомата и периодичного по построению автомата-наблюдателя, можно верифицировать, выбирая значения параметров согласно утверждению 4. Рассуждения для утверждения 3 аналогичны.

Таким способом были верифицированы разработанные автором модели функциональной задачи (период равен периоду задачи) и планировщика ядра (период равен интервалу планирования).

б) Непериодические автоматы.

Рассмотрим случай, когда в автомате используется только один временной параметр p. В этом случае имеет место следующее утверждение.

Утверждение 5. Пусть параметризованный автомат имеет несколько таймеров, в т.ч. хотя бы один неостанавливаемый таймер, и один временной параметр p. Пусть в сравнениях с таймерами может использоваться только параметр p, и, возможно, константа 0. Тогда для любой локации автомата верно, что если она не достижима при $p = 0$ и $p = 1$, то она не достижима при любых целых неотрицательных значениях параметра p.

Доказательство.

Докажем утверждение 5 методом от противного. Предположим, что для некоторого параметризованного автомата, удовлетворяющего условиям настоящего утверждения, и некоторой локации l этого автомата верно, что эта локация недостижима при $p = 0$ и $p = 1$ и достижима при $p = k, k > 1$.

Рассмотрим некоторый экземпляр A^k исходного параметризованного автомата, для которого параметр p имеет значение k, и для которого достижима локация l. Также рассмотрим экземпляр A^1 исходного параметризованного автомата, отличающийся от A^k лишь тем, что для него параметр p имеет значение 1.

Согласно предположению, для автомата A^k существует вычисление S^k, содержащее переход в локацию l. Докажем, что для автомата A^1 существует вычисление S^1, также содержащее

переход в локацию l. Для этого построим вычисление S^1 на основе вычисления S^k. Начальные состояния вычислений S^k и S^1 совпадают. S^1 будем строить так, чтобы для каждого состояния s^k в вычислении S^k и соответствующего этому состоянию состояния s^1 в вычислении S^1 было верно, что эти состояния различаются только значениями таймеров, и при этом значение каждого таймера в s^1 в k раз меньше значения этого таймера в s^k. Для начальных состояний обоих вычислений это верно по определению начального состояния.

Рассмотрим последовательно каждый переход вычисления S^k, вплоть до перехода в локацию l. Переход имеет один из двух видов:

1. Непрерывный переход, т.е. изменение значений таймеров с $\overline{c^k}$ до $\overline{c^k}'$, где ${c^k}_i' = {c^k}_i + \delta^k$, если в текущей локации i-й таймер активен, и ${c^k}_i' = {c^k}_i$ иначе. Помещаем в S^1 непрерывный переход, т.е. изменение значений таймеров с $\overline{c^1}$ до $\overline{c^1}'$, соответствующее увеличению значений всех активных таймеров на значение δ^1, такое, что $\delta^1 = \delta^k/k$. Для сформированного таким образом перехода в S^1 верно, что в состоянии, следующем за выполнением этого перехода в S^1, значение каждого таймера в k раз меньше, чем значение этого таймера в соответствующем состоянии S^k (поскольку это верно для состояний, предшествующих переходу, и $\delta^1 = \delta^k/k$). Следовательно, для состояний, следующих за выполнением рассматриваемого перехода в S^1 и S^k, значение каждого сравнения вида $t\ op\ 1$ в S^1 совпадает со значением соответствующего сравнения вида $t\ op\ k$ в S^k, где op — одна из стандартных операций сравнения.

   Обоснуем, что такой переход возможен в S^1. Поскольку в состоянии, предшествующем выполнению рассматриваемого перехода в S^1, значение каждого таймера в k раз меньше, чем значение этого таймера в соответствующем состоянии S^k, то значение каждого сравнения вида $t\ op\ 1$ в S^1 совпадает со значением соответствующего сравнения вида $t\ op\ k$ в S^k. Этот факт имеет место и для состояний, следующих за выполнением перехода. Следовательно, значения инвариантов текущих локаций в S^1 и S^k также совпадают как до выполнения перехода, так и после (т.к. совпадают текущие локации и значения всех сравнений таймеров). Поэтому рассматриваемый переход возможен в S^1.
2. Дискретный переход. Соответствующий переход возможен в S^1, т.к. в состояниях, предшествующих рассматриваемому переходу, значение каждого сравнения вида $t\ op\ 1$ в S^1 совпадает со значением соответствующего сравнения вида $t\ op\ k$ в S^k (а, следовательно, если условие перехода и инвариант локации назначения истинны в S^k, то они истинны и в S^1). Добавляем соответствующий переход в S^1 без изменений. Единственным возможным изменением значений таймеров при рассматриваемом переходе является обнуление. При этом по построению в рассматриваемых переходах обоих вычислений обнуляются одни и те же таймеры. Поэтому после выполнения этого перехода значение каждого таймера в S^1 в k раз меньше значения этого таймера в S^k.

По описанной схеме получим вычисление S^1 автомата A^1, содержащее переход в локацию l. Следовательно, предположение о том, что эта локация недостижима при $p = 0$ и $p = 1$ и достижима при $p = k, k > 1$, неверно. Утверждение 5 доказано.

Рассмотрим условия утверждения 5 в применении к автомату, эквивалентному сети автоматов из автомата-модели и автомата-наблюдателя, с учетом правил построения такого автомата (см.

стр. 140) и определения автомата-наблюдателя (не имеет собственных параметров). Эти условия представляют собой ограничение снизу на количество неостанавливаемых таймеров (в автомате должен быть хотя бы один такой таймер), ограничение на количество временных параметров (допустим ровно один такой параметр) и ограничения на вид элементарных сравнений таймеров. Из выполнения этих условий для каждого из двух автоматов сети следует их выполнение для автомата, эквивалентного этой сети. Поэтому, если условия утверждения 5 выполнены для автомата-модели и автомата-наблюдателя, то для проверки достижимости «плохой» локации в автомате-наблюдателе при произвольных значениях параметра p достаточно проверить достижимость этой локации при $p = 0$ и $p = 1$.

В соответствии с утверждением 5 был верифицирован автомат, моделирующий виртуальный канал.

Для непериодических автоматов, имеющих более одного параметра, автору не известно способов сокращения множества значений параметров, перебираемых при верификации. Таких автоматов среди разработанных в рамках данной работы нет. Однако, в случае необходимости включения в модель непериодического автомата с несколькими временными параметрами можно выполнить декомпозицию этого автомата на насколько более простых, каждый из которых верифицируется отдельно.

Согласно утверждениям 3 и 4, таймеры разрешено сравнивать лишь со значениями параметров, и обнулять только синхронно. Однако в автоматах-наблюдателях зачастую требуется проверить мгновенность реакции автомата на некоторое событие. Для этого может быть использован специальный таймер t^0, который нельзя останавливать, допустимо сравнивать лишь с константой 0 и можно обнулять не синхронно с другими таймерами в произвольные моменты времени. В отличие от таймера, который сравнивается с параметром p, и обнуление/запуск которого в момент x может привести к появлению новой границы региона, равной $x + p$ (в терминах утверждения 4 — нового возможного момента $x+p$ дискретного перехода), обнуление/запуск таймера t^0 никогда не приводит к появлению новых границ регионов (новых моментов дискретных переходов). Поэтому добавление таких таймеров в модель не ограничивает применения утверждений 3 и 4.

Б.3. Параметры, задающие размеры массивов

Согласно предложенной в разделе 2.4 обобщенной модели, параметризованные автоматы, входящие в модель МВС и реализующие базовые типы автоматов **`TS`**, **`T`** и **`CS`**, работают с массивами параметров, переменных и каналов. Размеры этих массивов являются параметрами соответствующих автоматов. Выделенные автором требования корректности к моделям инвариантны к размерам массивов. Модели должны быть реализованы таким образом, чтобы требования корректности выполнялись при любых размерах массивов. Сформулируем такие ограничения на автоматы, работающие с массивами (как автоматы-модели, так и автоматы-наблюдатели), что выполнения этих ограничений достаточно для того, чтобы из недостижимости «плохой» локации

на всех массивах размера, не большего некоторого k, следовала недостижимость этой локации при любых размерах массивов. Это позволит при верификации ограничиться рассмотрением массивов размера, не превышающего k. Также сформулируем правила определения этого k. Вопрос выбора достаточных для верификации множеств *значений* элементов массивов в данном пункте не рассматривается. Если элементы массивов являются временными параметрами, то для них выбор этих множеств значений выполняется согласно рассуждениям раздела Б.2, если переменными интерфейса — раздела Б.5. Также элементы массивов могут быть каналами; перебор всех возможных последовательностей синхронизаций по этим каналам при верификации осуществляется согласно правилам построения автомата-наблюдателя (во всех локациях автомата-наблюдателя активны переходы с действиями синхронизации, парными ко всем возможным действиям синхронизации исходного автомата).

Переменные, которые могут использоваться для обращения к элементу массива по индексу с помощью оператора [], будем называть *индексными*.

В рассуждениях данного пункта используется как оператор проверки на равенство, так и оператор присваивания. В соответствии с синтаксисом UPPAAL, оператор проверки на равенство будет обозначаться знаком «==», а оператор присваивания — знаком «=». Нумерация элементов массивов начинается с 0.

Введем класс параметризованных автоматов, работающих с массивами $a\_1, ..., a\_p$, и удовлетворяющих следующим условиям:

1. Размер всех массивов одинаков и равен параметру n.

2. Параметр n используется только внутри функций описанных ниже видов а), б), в). Каждая функция работает с одним или несколькими массивами переменных интерфейса, при этом работа с несколькими массивами осуществляется таким образом, что в циклах for на i-й итерации выполняются действия и проверки над i-ми элементами массивов.

Пусть $cond(i)$ — функция с логическим значением, в которой значения i-х элементов массивов могут сравниваться с числовыми константами и между собой, либо функция, возвращающая константу $true$. Будем говорить, что для элементов с индексом i выполняется условие $cond$, если функция $cond(i)$ возвращает значение $true$.

а) Для определения значения, возвращаемого функциями вида а), введем вспомогательное множество I_q, которое может быть получено по следующей схеме:

(i) выбирается множество индексов I_0, для которых выполняется условие $cond$; если $I_0 = \varnothing$, возвращается -1;

(ii) выбирается $I_1 \subseteq I_0$ такое, что в массиве a_1 значения элементов с индексами из I_1 являются наибольшими/наименьшими среди значений элементов с индексами из I_0;

(iii) выбирается $I_2 \subseteq I_1$ такое, что в массиве a_2 значения элементов с индексами из I_2 являются наибольшими/наименьшими среди значений элементов с индексами из I_1;

(iv) ...

(v) выбирается $I_q \subseteq I_{q-1}$ ($q \leqslant p$) такое, что в массиве a_q значения элементов с индексами из I_q являются наибольшими/наименьшими среди значений элементов с индексами из I_{q-1}.

Функции вида а) возвращают некоторый элемент множества I_q, полученного по описанной схеме. При этом для конкретной функции на каждом шаге (ii)—(v) используется фиксированный вариант выбора (выбираются либо наибольшие, либо наименьшие значения), но на разных шагах использующиеся варианты выбора могут быть разными.

Далее будем рассматривать только такие наборы массивов, что в массивах a_q, используемых на последнем этапе выбора индекса, значения всех элементов различны. Для этого должны быть различны начальные значения элементов массива, и все действия над массивом в автомате должны сохранять свойство различности значений элементов массивов a_q. Для таких массивов a_q возвращаемое функцией значение определяется однозначно. В разных функциях вида а) в автомате роль массива a_q могут играть разные массивы, и количество обрабатываемых разными функциями вида а) массивов может быть разным.

Функция, возвращающая один из элементов множества I_q, может быть реализована следующим кодом:

```
1.    int fa()
2.    {
3.        int res = -1;
4.        for (i : int[0, n-1]) {
5.            if (cond(i)) {
6.                if (res == -1 || (a_1[i] op_1 a_1[res]))
7.                    res = i;
8.            }
9.        }
10.       if (res == -1)
11.           return res;
12.       for (i : int[0, n-1]) {
13.           if (cond(i)) {
14.               if ((a_1[i] == a_1[res]) && (a_2[i] op_2 a_2[res]))
15.                   res = i;
16.           }
17.       }
18.       ...
19.       for (i : int[0, n-1]) {
20.           if (cond(i)) {
21.               if ((a_1[i] == a_1[res]) && (a_2[i] == a_2[res]) && ...
22.                   && (a_q[i] op_q a_q[res]))
23.                   res = i;
24.           }
25.       }
26.       return res;
27.   }
```

$op\_1, op\_2, ..., op\_q$ — операции сравнения $\{<\ ,\ >\}$; op_i и op_j могут различаться для различных $i,j\ \in\ \overline{1,q}$.

б) Функция, возвращающая значение $true$, если для любого $i\ \in\ \overline{0,n-1}$ для элементов с индексом i выполняется условие $cond$, и $false$ иначе:

```
1.    bool fb()
2.    {
3.        for (i : int[0, n-1]) {
4.            if (cond(i))
5.                return false;
```

```
6.          }
7.          return true;
8.      }
```

в) Функция, при выполнении которой всем элементам всех или некоторых массивов присваиваются константные значения (при этом для всех элементов одного массива используется одна и та же константа):

```
1.      void fc()
2.      {
3.          for (i : int[0, n-1]) {
4.              a_1[i] = x_1;
5.              ...
6.              a_p[i] = x_p;
7.          }
8.      }
```

$x\_1, ..., x\_p$ — числовые константы. Если в автомате используются функции вида а), то в функциях вида в) не изменяются элементы массивов a_q (т.е. тех массивов, которые используются на последнем этапе выбора индекса в функциях вида а), и элементы которых уникальны).

3. Значения, возвращаемые функциями вида а), присваиваются только индексным переменным.

4. Помимо оператора [], индексные переменные могут использоваться только в сравнениях вида $i\ op\ j$, где i — индексная переменная, j — индексная переменная или число -1, op — один из операторов «==» и «!=», а также в операторе $=$ (при этом в левой части оператора $=$ может быть только индексная переменная, а в правой — либо функция вида а), либо другая индексная переменная).

5. Функции вида а) могут использоваться только в операторах сравнения с индексными переменными, либо в правых частях оператора $=$ (левая часть в этом случае должна быть индексной переменной).

6. В операторе [] может использоваться только либо индексная переменная, либо функция вида а).

7. Из любой локации автомата имеется безусловный переход в нее же, при котором выполняется произвольное изменение переменных-элементов массивов (при этом сохраняется свойство уникальности элементов массивов a_q, использующихся на последнем этапе выбора индекса в функциях вида а)). Другие действия при таком переходе не выполняются.

8. Начальные значения индексных переменных равны -1. Если в автомате используются функции вида а), то в каждом массиве a_q начальные значения элементов различны.

9. Если при выполнении перехода выполняются действия, включающие в себя вызовы функций вида в) (изменение элементов массивов), то все такие вызовы выполняются после вызовов функций видов а) и б) и изменения индексных переменных.

10. Изменение элементов массивов выполняется только либо на переходах вида 7, либо внутри функций вида в).

11. Инварианты локаций не содержат обращений к каким-либо функциям, а также сравнений, включающих индексные переменные и элементы массивов.

Назовем такой класс автоматов У6 (для него далее будет сформулировано утверждение 6).

Автомат, удовлетворяющий условию 7, получается при построении автомата, эквивалентного сети из автомата-модели и автомата-наблюдателя, согласно правилам, приведенным на странице 140, так как:

- по определению автомата-наблюдателя из любой его локации имеется безусловный переход в нее же, при котором выполняется произвольное изменение элементов массивов (с учетом ограничений на допустимые значения этих элементов, в т.ч. с сохранением уникальности элементов массивов a_q в случае использования функций вида а));
- если в одном из двух автоматов сети (в данном случае — в автомате-наблюдателе) имеется безусловный переход без синхронизаций из некоторой локации L в нее же с выполнением некоторых действий, то в эквивалентном этой сети автомате для каждой локации L', соответствующей локации L рассматриваемого автомата, будет иметься безусловный переход без синхронизаций в нее же с выполнением тех же самых действий;
- каждая локация автомата, эквивалентного рассматриваемой сети автоматов, соответствует некоторой локации автомата-наблюдателя.

Недостижимость «плохой» локации в автомате-наблюдателе эквивалентна недостижимости всех соответствующих ей локаций в автомате, эквивалентном сети из исходного автомата и автомата-наблюдателя.

В доказательстве сформулированного ниже утверждения 6 используется понятие перестановки. Под перестановкой для массивов размера n будем понимать набор пар $\{(0, i_0), ..., (n-1, i_{n-1})\}$, таких что $\forall k \in \overline{0, n-1}$, верно, что $i_k \in \overline{0, n-1}$; и $\forall k_1, k_2 \in \overline{0, n-1}, k_1 \neq k_2$, верно, что $i_{k_1} \neq i_{k_2}$. При выполнении перестановки 0-й элемент массива ставится на позицию i_0, 1-й — на позицию i_1, и т.д., $(n-1)$-й — на позицию i_{n-1}. Под согласованной перестановкой понимается такая перестановка, которая применяется сразу ко всем массивам автомата: если i-й элемент одного массива перемещается в результате этой перестановки на j-ю позицию, то и во всех остальных массивах i-е элементы перемещаются на j-е позиции.

Если функция вида а) возвращает значение j, не равное -1, то для любой согласованной перестановки, содержащей элемент (j, i), после выполнения этой перестановки рассматриваемая функция станет возвращать значение i. При этом *все* функции вида а), возвращавшие до выполнения перестановки значение j, после выполнения перестановки станут возвращать значение i. Функции вида а), возвращавшие до выполнения перестановки значения, отличные от j, после выполнения перестановки станут возвращать значения, отличные от i. Это верно, т.к. выполнение условия $cond$ для элемента массива не зависит от его положения в массиве, и наибольший или наименьший элемент остается таковым независимо от своего положения в массиве. Однозначность определения возвращаемого индекса обеспечивается ограничением на входные данные (все элементы массивов a_q изначально различны) и на модификацию массивов в функциях вида в) (элементы массивов a_q не изменяются), а также запретом на изменение элементов массивов вне функций (кроме недетерминированного изменения, сохраняющего уникальность элементов массивов a_q — см. условия 7 и 10 определения класса У6).

Функции вида б) проверяют выполнение некоторого условия для элементов массивов со всевозможными номерами (условие не зависит от номера элементов) и нечувствительны к согласованным перестановкам элементов массивов.

Результат выполнения функции вида в) не зависит от порядка элементов в массиве, так как всем элементам одного и того же массива присваивается значение одной и той же константы. Важно, что после выполнения функции такого вида условие $cond$ либо выполняется для элементов массивов со всеми номерами, либо не выполняется для элементов ни с каким номером.

Утверждение 6. Для любой локации автомата, принадлежащего классу У6, верно, что если эта локация не достижима при любых размерах массивов, не превышающих $k + m$, где k — количество используемых индексных переменных, m — количество функций вида а), то эта локация не достижима для массивов любого размера. Количество функций видов б) и в) при этом несущественно.

Замечание. Подход к сокращению множества перебираемых *значений* элементов массивов описан далее в разделе Б.5 данного приложения.

Доказательство.

Докажем утверждение 6 методом от противного. Предположим, что для некоторого автомата, имеющего k индексных переменных и m функций вида а), выполнены все указанные условия, и существует локация E, недостижимая при любых размерах массивов, не превышающих $k + m$, но достижимая при использовании массивов размера $k + m + l$ для некоторого $l > 0$.

Рассмотрим вычисление S некоторого экземпляра данного параметризованного автомата (т.е. вычисление автомата без параметров), содержащее переход в локацию E. В этом автомате используются массивы размера $k + m + l$. По этому вычислению построим другое вычисление S' того же автомата, также содержащее переход в локацию E, такое что на всех его переходах значения индексных переменных и функций не превышают $k + m - 1$.

Построение S' будет основано на добавлении в частично построенное вычисление S' переходов, соответствующих переходам вычисления S, а также переходов с выполнением согласованных перестановок элементов массивов. Выполнение таких перестановок является частным случаем произвольного изменения элементов массивов, а поэтому переход с выполнением перестановок возможен из любого состояния автомата, согласно условию 7 (здесь и далее в доказательстве используются нумерованные условия определения класса автоматов У6). При этом перестановки сохраняют уникальность элементов массивов a_q. Для краткости вместо слов «согласованная перестановка» далее используется просто слово «перестановка».

Состояние автомата определяется его текущей локацией и совокупностью значений его переменных и таймеров. Поскольку, согласно условиям 9 и 10, для любого перехода значения элементов массивов при вызове и выполнении функций видов а) и б) совпадают со значениями в состоянии, предшествующем этому переходу, то далее будем рассматривать только значения функций в состояниях вычисления, а значения функций на переходах отдельно рассматриваться не будут. Из условий 4 и 9 также следует, что если индексная переменная изменяется при выполнении перехода, то ее новое значение равно значению либо некоторой функции, либо некоторой индексной переменной в состоянии, предшествующем переходу. Следовательно, если в состоянии, предшествующем переходу, значения всех функций и индексных переменных не превышают

$k+m-1$, то на переходе и в состоянии, следующем на ним, значения *индексных переменных* не превышают $k+m-1$ (даже в случаях, когда при выполнении перехода значения индексных переменных могут меняться несколько раз). Поэтому далее будут рассматриваться значения индексных переменных в состояниях, а их значения на переходах отдельно рассматриваться не будут.

Будем рассматривать последовательно каждый переход в вычислении S и строить вычисление S' по вычислению S. В процессе построения текущим состоянием вычисления S будем называть начальное состояние рассматриваемого перехода в S, а текущим состоянием вычисления S' — конечное состояние последнего добавленного перехода в S', либо начальное состояние S', если ни один переход еще не был добавлен в S'. На каждом шаге построения S' (один шаг соответствует рассмотрению одного перехода из S) для текущих состояний S и S' должны выполняться следующие свойства (♣):

(i) В текущем состоянии вычисления S' значения индексных переменных и функций вида а) меньше $k+m$.

(ii) Значения каждого логического выражения вида $A\ \ op\ \ B$ (A, B — функция или индексная переменная, op — один из операторов «==» и «!=») совпадают в текущем состоянии вычисления S и в текущем состоянии вычисления S'.

(iii) Для любой функции/индексной переменной верно, что если эта функция/индексная переменная имеет в текущем состоянии вычисления S значение i, а в текущем состоянии вычисления S' — значение j, то для любого используемого в автомате массива a верно, что значение элемента $a[i]$ в текущем состоянии вычисления S совпадает со значением элемента $a[j]$ в текущем состоянии вычисления S'.

(iv) Текущие состояния вычислений S и S' могут различаться только значениями индексных переменных и *порядком следования* значений элементов массивов, т.е. массивы в этих состояниях совпадают с точностью до согласованной перестановки.

При условии выполнения свойств ♣, для любого логического выражения, используемого в рассматриваемом автомате, верно, что его значение в текущем состоянии вычисления S совпадает со значением в текущем состоянии вычисления S', так как для этих состояний:

– Значения сравнений, в которых используются функции вида а) и индексные переменные, совпадают согласно свойству (ii) ♣ и условиям 4 и 5, накладывающим ограничения на использование функций вида а) и индексных переменных.
– Значения сравнений, в которых используются элементы массивов, совпадают, согласно свойству (iii) ♣ и условию 6, накладывающему ограничение на конструкции для доступа к элементам массивов.
– Значения сравнений, в которых не используются индексные переменные, элементы массивов и функции вида а), совпадают согласно свойству (iv) ♣.
– Значения функций вида б) не зависят от порядка следования значений элементов массивов и значений индексных переменных, а рассматриваемые состояния могут различаться только порядком следования значений элементов массивов и значениями индексных переменных, согласно свойству (iv) ♣.
– В функциях, возвращающих логическое значение и отличных от функций вида б), могут использоваться сравнения только приведенных в данном списке видов.

Таким образом, рассмотрены все «элементарные» логические выражения, которые могут использоваться для формирования произвольных логических выражений, и показано, что их значения совпадают для текущих состояний S и S' при условии выполнения свойств ♣. Значит, если для текущих состояний S и S' выполнены свойства ♣, то любое условие перехода, истинное в текущем состоянии S, также истинно и в текущем состоянии S'. Из того, что текущие состояния в S и S' различаются только значениями индексных переменных и порядком следования значений элементов массивов (свойство (iv) ♣), для этой пары состояний значения каждого выражения, в котором используются элементы массивов, совпадают (свойство (iii) ♣ и условие 6), индексные переменные и функции изолированы от других переменных (условия 4 и 5), значения всех логических выражений совпадают, следует, что если из текущего состояния S будет выполнен переход с некоторыми действиями, и из текущего состояния S' будет выполнен переход с теми же самыми действиями, то полученные состояния вычислений S и S' будут различаться только значениями индексных переменных и элементов массивов. Поскольку никакие функции и значения индексных переменных и элементов массивов не используются в инвариантах локаций (условие 11), то для полученной новой пары состояний значения инвариантов локаций будет совпадать.

Следовательно, если для некоторой рассматриваемой пары состояний в S и S' выполняются условия ♣, то любой переход, возможный из рассматриваемого состояния в S, возможен и из соответствующего состояния в S'.

Рассмотрим начальное состояние S. Значения всех индексных переменных в этом состоянии равны -1 (согласно условию 8).

Если значения всех функций вида а) в начальном состоянии S меньше $k+m$, то в качестве начального состояния S' возьмем начальное состояние S. В этом случае выполнение свойств ♣ для начальных состояний вычислений S' и S непосредственным образом следует из видов этих состояний.

Если значения некоторых функций вида а) в начальном состоянии S не меньше $k+m$, то сформируем начальное состояние S' следующим образом. Сформируем перестановку элементов массивов. Сначала рассмотрим все числа $w_0, ..., w_h$, $h \leqslant m-1$, возвращаемые функциями вида а) в начальном состоянии S. Для каждого такого числа w_i добавим в формируемую перестановку элемент (w_i, i). Далее для каждого числа w от 0 до $n-1$, кроме рассмотренных ранее (т.е. не возвращаемого ни одной из функций вида а) в начальном состоянии S), добавим в формируемую перестановку элемент (w, n_w), где n_w — первое целое неотрицательное число, не использованное ранее как второй компонент формируемых элементов перестановки. Перестановка полностью сформирована. Она содержит ровно n элементов, при ее построении в качестве первых компонентов формируемых элементов перестановки в совокупности рассмотрены все возможные номера элементов массивов от 0 до $n-1$, а в качестве вторых компонентов в совокупности рассмотрены все числа от 0 до $n-1$. Поэтому полученная перестановка является корректной. Применим сформированную перестановку к массивам начального состояния S. В качестве начального состояния S' возьмем состояние, отличающееся от начального состояния S лишь значениями элементов массивов. А в качестве значений элементов массивов для начального состояния S' возьмем значения, полученные в результате выполненной перестановки. Итого, для начальных состояний S и S' выполняются свойства ♣. Условия (i—iii) выполнены по построению начального состояния S' и по

построению перестановки: каждая функция, возвращающая в начальном состоянии S значение w_i, в начальном состоянии S' возвращает значение i, $i < m$; при этом если $w_i \neq w_j$, то $i \neq j$. Условие (iv) выполнено по построению начального состояния S'.

Итого свойства ♣ для начальных состояний S и S' выполняются. Далее будем обосновывать их выполнение по мере описания шага построения S'.

Любой переход в S имеет один из видов: переход вида, описанного в условии 7 (частный случай дискретного перехода); дискретный переход отличного от описанного в условии 7 вида; непрерывный переход. Таким образом, для очередного рассматриваемого перехода вычисления S имеет место один из следующих случаев:

1. Переход вида, описанного в условии 7 — безусловный переход из некоторой локации в нее же с некоторым изменением значений элементов массивов (с сохранением уникальности значений элементов массивов a_q). Другие составляющие состояния автомата при этом не меняются.

Если добавить в S' этот переход, то свойства (i)—(iii) ♣ для конечного состояния добавленного перехода в S' и конечного состояния рассматриваемого перехода в S могут нарушаться. Сформируем такую перестановку элементов массивов, чтобы после добавления в S' перехода с выполнением этой перестановки (как частного случая произвольного изменения элементов массивов), для конечного состояния рассматриваемого перехода в S и конечного состояния построенной части вычисления S' выполнялись все свойства ♣.

Рассмотрим каждый элемент массива и определим его позицию в формируемой перестановке.

1.1. Сначала рассмотрим все числа, возвращаемые функциями вида а) в конечном состоянии рассматриваемого перехода в S, и равные в этом состоянии S значениям некоторых индексных переменных. Таких различных чисел не больше, чем $min(k,m)$. Для каждого такого числа w получим значение w' той же индексной переменной в конечном состоянии последнего добавленного перехода в S'. Это значение и будет определять в формируемой перестановке элемент (w, w'), соответствующий числу w, рассматриваемому как индекс. Если значение функции совпадает со значениями нескольких индексных переменных в конечном состоянии рассматриваемого перехода в S, то значения этих индексных переменных совпадают и в конечном состоянии последнего добавленного перехода в S', т.к. до выполнения рассматриваемого перехода в S выполнялось условие (ii) ♣ и значения индексных переменных в результате выполнения рассматриваемого перехода в S не изменяются. Например, если функция f возвращает в конечном состоянии рассматриваемого перехода в S значение 100, индексная переменная x в этом состоянии S имеет значение 100, а в конечном состоянии последнего добавленного перехода в S' индексная переменная x имеет значение 5, то в формируемую перестановку необходимо добавить элемент $(100, 5)$ (при выполнении перестановки в каждом массиве элемент с номером 100 должен занять 5-е место). Таким образом обеспечивается то, что после добавления в S' перехода, соответствующего выполнению в массивах формируемой перестановки, для функций вида а), значения которых совпадают со значениями индексных переменных в конечном состоянии рассматриваемого перехода в S, их значения будут совпадать со значениями тех же индексных переменных и в конечном состоянии этого добавляемого перехода в S'. При этом в конечном состоянии этого добавляемого перехода в S' эти значения будут меньше $k + m$, т.к. для начального состояния рассматриваемого перехода в S и конечного

состояния последнего добавленного перехода в S' условие (i) ♣ выполнялось. В результате добавления в S' перехода, соответствующего выполнению в массивах формируемой перестановки, для каждого массива значение элемента с индексом w в конечном состоянии рассматриваемого перехода в S будет совпадать со значением элемента с индексом w' в конечном состоянии этого добавляемого перехода в S'.

1.2. Далее рассмотрим все числа, возвращаемые функциями вида а) в конечном состоянии рассматриваемого перехода в S, и не равные в этом состоянии S значениям никаких индексных переменных.

Для каждого такого числа w добавим в формируемую перестановку элемент (w, n_w), где n_w — первое целое неотрицательное число, не равное значению ни одной из индексных переменных в конечном состоянии последнего добавленного перехода в S' и не использованное ранее (в рамках анализа рассматриваемого перехода) как второй компонент формируемых элементов перестановки.

При этом максимальное выбранное указанным образом n_w не превышает $k + m - 1$. Обоснуем это. На шаге 1.1 в качестве вторых компонентов формируемых элементов перестановки используются значения индексных переменных в конечном состоянии последнего добавленного перехода в S', а на шаге 1.2 — целые неотрицательные числа, не являющиеся значениями индексных переменных в этом состоянии в S'. Независимо от количества элементов перестановки, сформированных на шаге 1.1, количество целых неотрицательных чисел, равных значениям индексных переменных в рассматриваемом состоянии S', не превышает k, и равно k, когда значения всех этих переменных различны. На шаге 1.2 такие числа в качестве вторых компонентов формируемых элементов перестановки использовать нельзя. Поэтому для формирования элементов перестановок на шаге 1.2 понадобится *еще* столько целых неотрицательных чисел, сколько функций с различными значениями рассматривается на этом шаге. Максимальное количество целых неотрицательных чисел, использующихся на шаге 1.2 в качестве вторых компонентов формируемых элементов перестановки, равно m (это значение достигается, когда все функции автомата рассматриваются на шаге 1.2, а не 1.1, и имеют различные значения). Таким образом, поскольку на шаге 1.2 выбираются минимальные из не выбранных на шаге 1.1 целых неотрицательных чисел, то максимальное выбранное указанным образом n_w не превышает $k + m - 1$.

Такой способ формирования элементов перестановки обеспечивает то, что в результате добавления в S' перехода, соответствующего выполнению в массивах формируемой перестановки, для функций, значения которых не совпадают ни с одним из значений индексных переменных в конечном состоянии рассматриваемого перехода в S, их значения не будут совпадать со значениями каких-либо индексных переменных и в конечном состоянии этого добавляемого перехода в S'. При этом в конечном состоянии этого добавляемого перехода в S' значения этих функций будут меньше $k + m$. Также для каждого массива значение элемента с индексом w, рассмотренным на шаге 1.2, в конечном состоянии рассматриваемого перехода в S будет совпадать со значением элемента с индексом n_w в конечном состоянии добавляемого перехода в S'.

Поскольку значения функций, рассмотренных на шаге 1.1, в конечном состоянии добавляемого перехода в S' определяются только элементами перестановки, сформированными на шаге

1.1, то добавление в перестановку новых элементов на шаге 1.2 не влияет на значения функций, рассмотренных на шаге 1.1, в конечном состоянии этого добавляемого перехода в S'.

1.3. Для каждого числа w от 0 до $n-1$, кроме рассмотренных на шагах 1.1, 1.2 (т.е. не возвращаемого ни одной из функций вида а) в конечном состоянии рассматриваемого перехода в S), добавим в формируемую перестановку элемент (w, n_w), где n_w — первое целое неотрицательное число, не использованное на шагах 1.1 и 1.2 как второй компонент формируемых элементов перестановки. Числа n_w на данном шаге могут превышать $k+m-1$, но они не равны ни одной из индексных переменных и не возвращаются ни одной функцией вида а) в конечном состоянии добавляемого перехода в S', а следовательно не используются в условиях и действиях никаких переходов из этого состояния.

Необходимая перестановка полностью сформирована. Она содержит n элементов. На шагах 1.1—1.3 в совокупности рассмотрены все возможные номера элементов массивов. На шаге 1.1 в качестве вторых компонентов формируемых элементов перестановки выбирались *различные* числа, не превышающие $k+m-1$. На шагах 1.2, 1.3 при формировании элемента перестановки для каждого номера в качестве второго компонента выбиралось минимальное из не использованных ранее (для предыдущих номеров) целых неотрицательных значений. Поэтому полученная перестановка является корректной.

В конечном состоянии добавляемого перехода в S' значение каждой функции вида а) определяется только соответствующим этой функции элементом перестановки (т.е. элементом перестановки, сформированным на шаге 1.1 или 1.2 в соответствии со значением этой функции в конечном состоянии рассматриваемого перехода в S), но не другими элементами перестановки. Поэтому при формировании перестановки добавление в нее новых элементов не влияет на выполнение в конечном состоянии добавляемого перехода в S' свойств, которые обеспечиваются ранее добавленными элементами перестановки.

По построению перестановки, для каждого массива значения элементов этого массива с номерами, определяемыми некоторой индексной переменной, совпадают в конечном состоянии рассматриваемого перехода в S и в конечном состоянии добавляемого перехода в S'. Аналогично для элементов массива с номерами, определяемыми некоторой функцией вида а). Т.е. для этих двух состояний вычислений S и S' выполняется свойство (iii) ♣. Свойства (i), (ii) ♣ выполняются для этих состояний также по построению перестановки (их выполнение обеспечивают действия шагов 1.1 и 1.2). Свойство (iv) выполняется, т.к. оно выполнялось для начального состояния рассматриваемого перехода в S и конечного состояния последнего добавленного перехода в S', и в результате выполнения рассматриваемого (в S) и добавляемого (в S') переходов меняются только значения элементов массивов.

Таким образом, по построению после выполнения этой перестановки, для конечного состояния рассматриваемого перехода в S и конечного состояния добавляемого перехода в S' будут выполнены все свойства ♣.

Добавляем в S' дискретный переход с изменением элементов массива, соответствующим выполнению перестановки, и переходим к следующему шагу, то есть к следующему переходу в S.

2. Дискретный переход отличного от описанного в условии 7 вида. При таком переходе может происходить смена локации и изменение значений переменных и таймеров.

Дискретный переход в вычислении S' из текущего состояния S', для которого (перехода) локация назначения, условие и действия совпадают с локацией назначения, условием и действиями некоторого дискретного перехода T в вычислении S из текущего состояния S, будем называть переходом, *соответствующим* этому дискретному переходу T.

Так как на предыдущем шаге построения S' выполнялись свойства ♣, то переход, соответствующий рассматриваемому переходу в S, возможен и в S'.

Если рассматриваемый переход в S содержит вызовы функций вида в), то, в соответствии с условием 9, этот переход можно разбить на два последовательных перехода: (I) выполнение всех действий, кроме вызовов функций вида в), в т.ч. изменение индексных переменных и вызовы функций вида а), и (II) вызовы функций вида в). Если рассматриваемый переход в S не содержит вызовы функций вида в), то обозначим его как (I) (а переход (II) в этом случае отсутствует).

Добавляем переход в S' переход, соответствующий переходу (I) в S. Из того, что свойства ♣ выполняются для состояния, предшествующего выполнению перехода (I) в S, и состояния, предшествующего выполнению соответствующего перехода в S' (в т.ч. значения функций и индексных переменных до выполнения этого перехода в S' не превышают $k+m-1$), и при выполнении этих переходов не происходит изменения массивов, следует, что в состояниях, непосредственно следующих за выполнением этих переходов в S и S', свойства ♣ по-прежнему выполняются:

- Свойство (i) ♣ выполняется, т.к. если в результате выполнения добавленного перехода в S' были изменены значения некоторых индексных переменных, то новые значения этих переменных равны значениям других индексных переменных или функций в состоянии, предшествующем этому переходу в S' (в т.ч. для новых значений переменных будет выполнено свойство (i) ♣ в случае выполнения «цепочек» операторов присваивания для индексных переменных).
- Свойство (ii) ♣ выполняется, т.к. имеют место факты (1) и (2): (1) свойство (ii) ♣ выполняется для состояния, предшествующего переходу (I) в S, и состояния, предшествующего соответствующему переходу в S'; (2) если при выполнении этих переходов в S и S' изменяется некоторая индексная переменная, то ей присваивается значение одной и той же функции или другой переменной, и порядок выполнения присваиваний при выполнении этих переходов в вычислениях S и S' один и тот же.
- Свойство (iii) ♣ выполняется, т.к. имеют место факты (1), (2) и (3): (1) свойство (iii) ♣ выполняется для состояния, предшествующего переходу (I) в S, и состояния, предшествующего соответствующему переходу в S'; (2) значения элементов массивов при выполнении этих переходов не меняются (следовательно, не меняются и значения функций вида а)); (3) если при выполнении этих переходов в S и S' изменяется некоторая индексная переменная, то ей присваивается значение одной и той же функции или другой индексной переменной, и порядок выполнения присваиваний при выполнении этих переходов в вычислениях S и S' один и тот же.
- Свойство (iv) ♣ выполняется, т.к. имеют место факты (1), (2) и (3): (1) свойство (iv) ♣ выполняется для состояния, предшествующего переходу (I) в S, и состояния, предшествующего соответствующему переходу в S'; (2) индексные переменные «изолированы» от

других переменных (условия 4 и 5); (3) локация назначения и действия (в т.ч. по изменению не индексных переменных и таймеров) этих переходов в S и S' совпадают.

Если переход (II) отсутствует, то переходим к следующему шагу. Если имеется переход (II), то добавляем в S' переход, соответствующий переходу (II). В состоянии, следующем за выполнением этого добавленного перехода в S', значения индексных переменных по-прежнему не превышают $k+m-1$, однако значения, возвращаемые функциями, могут превышать $k+m-1$ за счет выполнения функций вида в). Поэтому, чтобы обеспечить выполнение условий ♣ на данном шаге, сформируем перестановку и добавим в S' переход с изменением элементов массивов, соответствующим ее выполнению. Перестановка формируется в точности так же, как и в случае 1. Различие лишь в том, что причиной появления среди возвращаемых функциями значений чисел, больших $k+m-1$, в случае 1 является произвольное изменение элементов массивов, а в случае перехода (II) — выполнение функций вида в). После добавления в S' перехода с полученным изменением элементов массивов, условия ♣ будут выполняться по построению перестановки (обоснование этого совпадает с обоснованием для случая 1).

3. Непрерывный переход (продвижение времени).

Непрерывный переход в вычислении S' из текущего состояния S', значения таймеров в конечном состоянии которого совпадают со значениями тех же таймеров в конечном состоянии некоторого непрерывного перехода T в вычислении S из текущего состояния S, будем называть переходом, *соответствующим* этому непрерывному переходу T.

Так как на предыдущем шаге построения S' выполнялись условия ♣, то переход, соответствующий рассматриваемому переходу в S, возможен и в S'. Добавим такой переход в S'. Условия ♣ после этого будут по-прежнему выполняться: условия (i)-(iii) ♣ не зависят от значений таймеров, а условие (iv) ♣ выполняется по построению добавленного перехода. Перейдем к следующему шагу построения S'.

Таким образом будем добавлять в S' переходы, построенные на основе переходов из S, до тех пор, пока в S не будет достигнута локация E. Локация E будет достигнута и в S', так как в S имеется дискретный переход в эту локацию, а каждому дискретному переходу в S соответствует переход или цепочка переходов в S', при этом локация конечного состояния дискретного перехода в S совпадает с локацией конечного состояния соответствующего перехода или цепочки переходов в S'.

Функции вида а) таковы, что если функция возвращает некоторое значение x для массивов длины y, то эта функция будет возвращать значение x и для массивов, полученных из исходных отбрасыванием последних $y-x-1$ элементов, т.е. всех элементов, следующих за элементом с номером x.

Во всех состояниях и переходах вычисления S' значения индексных переменных меньше $k+m$. Значения функций вида а) меньше $k+m$ во всех состояниях вычисления S' кроме, быть может, конечных состояний переходов, соответствующих дискретным переходам вида (II) в S. Однако из каждого такого состояния в S' выполняется безусловный переход с изменением элементов массивов, в конечном состоянии которого значения всех функций вида а) меньше $k+m$. Функции вида а) в этом переходе не используются. Таким образом, во всех переходах, где используются функции вида а), их значения меньше $k+m$. Всего в каждом массиве автомата $k+m+l$

элементов. Отбросим у всех массивов последние l элементов. Согласно рассуждениям выше, значения индексных переменных на каждом из переходов при этом не изменятся; значения функций вида а) не изменятся на каждом из переходов, на котором эти функции используются. Перестановка — не специальное действие в автомате, а лишь используемый в доказательстве утверждения 6 вспомогательный аппарат для описания начальных значений элементов массивов и значений, присваиваемых этим элементам при выполнении перехода с их произвольным изменением. Значит, несмотря на то, что в рамках построения S' значения некоторых элементов с номерами, меньшими $k+m$, «берутся» из элементов с номерами, большими или равными $k+m$, наличие в массиве элементов с номерами, большими или равными $k+m$, для присваивания таких значений не требуется, и эти элементы могут быть отброшены. Также отметим, что отбрасывание элементов с номерами, большими или равными $k+m$, не нарушает уникальности значений элементов массивов a_q. Следовательно, переход с произвольным изменением значений элементов массивов и сохранением уникальности значений элементов массивов a_q остается таковым и после отбрасывания элементов с номерами, большими или равными $k+m$. Другие переходы, имеющиеся в вычислении S', также остаются корректными после отбрасывания указанных элементов массивов, т.к. по построению вычисления S' эти элементы не используются (ни их значения, ни их номера) ни на одном из переходов.

Таким образом получим вычисление автомата, содержащее локацию E, для массивов размера $m+k$, что противоречит введенному предположению. Следовательно, предположение неверно и утверждение 6 доказано.

Для того, чтобы автомат, полученный согласно правилам со страницы 140 из автомата-модели и автомата-наблюдателя, принадлежал классу У6, достаточно, чтобы для каждого из этих автоматов выполнялись условия 1—6 и 8—11. Этот вывод непосредственно следует из вида условий: в частности, если в автомате-модели и автомате-наблюдателе используются только функции видов, описанных в условии 2, то и функции результирующего автомата имеют только такие виды, т.к. никаких других функций, кроме как функций из исходных автоматов, в результирующем автомате нет. Условие 7 выполняется по построению автомата-наблюдателя. При применении утверждения 6 к автомату, эквивалентному сети автоматов из этих двух автоматов, в качестве m и k берутся суммарные по двум автоматам количества функций и индексных переменных.

В автоматах класса У6, параметр-размер массивов может использоваться только внутри функций вида а), б), в). В том числе последовательный проход по всем элементам массивов может выполняться только внутри таких функций. Один вызов функции соответствует одному переходу в автомате. В рамках этих ограничений возможно моделировать такие шаги функционирования МВС как выбор задачи, работа которой должна быть поставлена на выполнение; проверка получения сообщений из всех виртуальных каналов, определенных для задачи; считывание сообщений из всех виртуальных каналов и т.п. При этом операции, производимые с каждым элементом массива, достаточно просты для помещения внутрь конструкции for, принадлежащей функции.

Однако существуют шаги функционирования МВС, моделирование которых требует выполнения более сложных операций с элементами массивов. Так обработка элементов массивов с одним значением индекса может включать в себя последовательные переходы между несколькими локациями, на которых (переходах) проверяются условия и выполняются действия с этими

элементами массивов. В т.ч. обработка элементов массивов с одним значением индекса может включать несколько синхронизаций по каналам. При этом каналы могут быть элементами массивов. Например, для каждого окна модель планировщика ядра должна посредством синхронизаций оповестить модель планировщика соответствующего раздела об открытии и закрытии окна. Каждая синхронизация требует выполнения отдельного перехода, а конструкция for внутри функции (в т.ч. вида а), б), в)) не предполагает выполнения переходов, поэтому моделирование таких операций согласно ограничениям утверждения 6 затруднительно. В данном случае (моделирование планировщика ядра) автомат работает с двумя массивами каналов и тремя массивами параметров. Все эти массивы имеют одинаковый размер.

Для прохода вне функций по всем элементам массива необходимо либо использовать вне функций параметр, задающий размер массива, либо хранить в массиве на последней позиции специальный «служебный» элемент, являющийся маркером конца массива, и при каждом переходе к следующему элементу массива проверять, не является ли этот элемент таким маркером. Наличие в автомате таких конструкций, в общем случае крайне отрицательно сказывается на возможности верификации этого автомата для произвольных значений параметра-размера массива. Рассмотрим простейший пример (см. рисунок Б.4). В приведенном автомате p — параметр, i — переменная, в условиях и действиях переходов используются лишь конструкции $i == p$, $i < p$, $i = 0$, $i + = 1$, необходимые для работы с массивами.

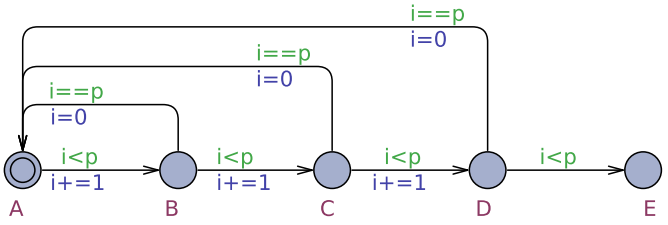

Рисунок Б.4 — Пример использования параметра вне функции

Локация E не достижима при $p \leqslant 3$ и достижима при любых $p > 3$. Достраивая автомат с рисунка Б.4 по аналогии, можно для любого заданного n получить автомат, в котором локация E не достижима при $p \leqslant n$ и достижима при любых $p > n$. Аналогичный пример имеет место и в случае использования маркера конца массива, если вместо конструкций $i == p$ и $i < p$ использовать соответственно конструкции $x[i] == MARKER$ и $x[i] != MARKER$. Таким образом, даже имея строгие ограничения на формат включающих параметр конструкций, но не вводя ограничения на структуру автомата (набор локаций и переходов между ними), нельзя гарантировать, что найдется такое число k, не зависящее от структуры автомата, что из выполнения требования к автомату при всех значениях параметра, не превышающих k, следует выполнение требования к автомату при любых значениях параметра.

Следовательно, для того, чтобы существовало такое число k, что при верификации автомата достаточно рассматривать массивы размера, не превышающего k, придется ввести ограничения на структуру автоматов, в которых параметр-размер массивов используется вне функций. Ограничения будут сформированы исходя из предположения, что автомат работает с несколькими массивами одного размера, обрабатывает элементы последовательно (при этом проход выполняется согласованно по всем массивам: сначала обрабатываются нулевые элементы всех массивов,

затем — первые и т.д.), и действия, выполняемые с элементами, не зависят от номеров этих элементов, кроме, быть может, элементов с номером 0, переход к которым также является переходом к следующему проходу по массивам. Под переходом к следующим элементам массивов понимается увеличение на 1 индексной переменной, использующейся для доступа к этим элементам по номеру, а под переходом к следующему проходу по массивам — присваивание этой индексной переменной значения 0. То есть если обработке элементов массивов с одинаковым номером $i > 0$ соответствует некоторая последовательность переходов в вычислении автомата, то аналогичная последовательность переходов соответствовала бы обработке элементов с теми же значениями, но другим номером $j > 0$ (под «значением» канала-элемента массива понимается сам канал). При этом количество переходов в этих двух последовательностях одинаково, а соответствующие друг другу состояния двух последовательностей различаются только значениями индексной переменной, используемой для прохода по массиву и, возможно, значениями таймеров. В данном случае подразумевается, что внешние воздействия (посредством синхронизаций по канала и изменения переменных интерфейса) на автомат при обработке элементов массивов с номером i совпадают с внешними воздействиями на него при обработке элементов массивов с номером j. Ограничения на вид автоматов будут введены формально и пояснены ниже.

Кроме того, требования к моделям компонентов МВС, представляющим собой такие автоматы, также должны быть инвариантны к размерам массивов и порядку элементов в массивах: требование может накладывать ограничения на последовательность синхронизаций, происходящих при обработке лишь элементов *с одним номером*, и на моменты этих синхронизаций (то есть значения не останавливаемого таймера). При этом ограничения на моменты выполнения синхронизаций, происходящих при обработке автоматом-наблюдателем элементов массивов с номером i, должны определяться значениями временных параметров, не являющихся элементами массивов, а также значениями временных параметров-элементов массивов с тем же номером i. По построению автомата-наблюдателя, параметры у автомата-наблюдателя и у автомата-модели общие. Иными словами, автомат-наблюдатель, соответствующий проверяемому требованию, также как и автомат-модель, должен обрабатывать элементы массивов последовательно, выполнять проход по всем массивам согласованно, и действия, выполняемые автоматом-наблюдателем с элементами массивов, не должны зависеть от номеров этих элементов (кроме, быть может, элементов с номером 0).

Сформируем более строго ограничения на вид параметризованного автомата, последовательно обрабатывающего элементы массива.

Для дальнейших рассуждений необходимо описать семантику конструкции $select$, используемой в UPPAAL для компактности описания автоматов. Для обозначения разметки перехода между локациями будем использовать строку вида « <конструкция select; условие; синхронизация; изменение переменных и таймеров> ». Строка разделена на четыре части символом «;». Каждая из частей может быть пустой. Например, «; ; ;» — безусловный переход без синхронизаций и изменения переменных и таймеров, «; $(t>0)$; $a!$; $x = 2$» — переход с условием $(t > 0)$, действием синхронизации $a!$ и действием $x = 2$. Разметка перехода из локации A в локацию B вида «$select\ i : [0,q-1]; cond(i); chan[i]!; act(i)$» означает, что из локации A в локацию B имеется q переходов: 1-й переход имеет условие $cond(0)$, действие синхронизации $chan[0]!$ и действия

по изменению переменных и таймеров $act(0)$; q-й переход — условие $cond(q-1)$, действие синхронизации $chan[q-1]!$ и действия по изменению переменных и таймеров $act(q-1)$. Запись «$cond(i)$» здесь обозначает некоторое логическое выражение, зависящее от i (и, возможно, от значений переменных, параметров и таймеров автомата); «$act(i)$» — последовательность действий по изменению переменных и обнулению таймеров, содержащая выражения, зависящие от i (и, возможно, от значений переменных, параметров и таймеров автомата). i в приведенной записи не входит в состояние автомата и не является переменной.

Введем класс автоматов У7, удовлетворяющих следующим условиям:

1. Автомат работает с набором массивов размера n, где n — параметр. Элементами массивов могут быть временные параметры и каналы (в каждом массиве что-то одно), но не переменные. Хотя бы один из массивов является массивом временных параметров, и хотя бы один из массивов является массивов каналов. Каналы, не являющиеся элементами массивов, отсутствуют.

2. В автомате имеются следующие внутренние переменные:

– i — индексная переменная с начальным значением -1;

– f — булева переменная с начальным значением 0 (является флагом-признаком того, что обрабатываются последние элементы массивов);

– $a\_1, ..., a\_m$ — остальные внутренние переменные с начальными значениями $x\_1, ..., x\_m$ (переменные $a\_1, ..., a\_m$ могут отсутствовать).

3. Переменные интерфейса отсутствуют.

4. Имеется не более одного неостанавливаемого таймера. Таймеры с возможностью останова не используются. Обнуление неостанавливаемого таймера происходит только внутри функций $start()$ и $next()$ вида, описанного в п.5.

5. Параметр n используется только в функциях $start()$ и $next()$, имеющих следующий вид:

```
1.    void start()
2.    {
3.        i=0;
4.        a_1=x_1; ... a_m=x_m;
5.        t=0;
6.        if (i==n-1)
7.            f=1;
8.        else
9.            f=0;
10.   }
```

```
1.    void next()
2.    {
3.        a_1=x_1; ... a_m=x_m;
4.        if (i==n-1) {
5.            i=0;
6.            t=0;
7.        } else
8.            i=i+1;
9.        if (i==n-1)
10.           f=1;
11.       else
12.           f=0;
13.   }
```

Приведенные функции в свою очередь используются только в двух переходах между тремя служебными локациями (обозначим эти локации как A, B и C). Эти переходы имеют следующий вид:

а) Источник: A. Назначение: B. Условие: $True$. Действия: $start()$.

б) Источник: C. Назначение: B. Условие: $True$. Действия: $next()$.

Переход вида а) соответствует началу работы с массивами. При его выполнении обнуляется значение таймера t, переменным присваиваются соответствующие начальные значения. Локация A является начальной локацией автомата.

Переход вида б) соответствует переходу к следующему номеру элементов массивов, либо к номеру 0, если последние элементы массивов обработаны. Если осуществляется переход к последним элементам массивов, то флаг f принимает значение 1.

Продвижение времени в локациях A, B и C запрещено. Переходы в локацию B, отличные от а) и б), отсутствуют. Переходы из локаций A и C, отличные от а) и б), отсутствуют. Переходы в локацию A также отсутствуют.

6. Единственной допустимой конструкцией, содержащей оператор «[]», является конструкция «$[i]$». Вне функций $start()$ и $next()$, в т.ч. в других функциях автомата, переменная i может использоваться только в конструкции «$[i]$».

7. Автомат функционирует в составе сети из двух автоматов, такой, что в любом состоянии этой сети для любого действия синхронизации, использующегося в рассматриваемом автомате, в другом автомате сети существует активный переход с действием синхронизации, парным для этого действия синхронизации. Для реализации таких переходов в парном автомате используется конструкция $select$, семантика которой описана подробно на стр. 179.

Это условие означает, что в рассматриваемом автомате любой активный переход, содержащий действие синхронизации, всегда может быть выполнен. Условие соответствует функционированию рассматриваемого автомата (модели) совместно с автоматом-наблюдателем.

8. Помимо служебных локаций A, B и C, в автомате имеется служебная локация D, такая что:

– В локацию D имеется безусловный переход из локации B. При выполнении действий этого перехода не используется и не изменяется значение таймера t. Другие переходы из локации B отсутствуют.
– В локации D разрешено продвижение времени и она имеет инвариант вида $t <= p[i]$, где p — один из массивов временных параметров.

Локация D соответствует ожиданию момента времени, в который должна начаться обработка элементов массивов с текущим номером.

Пример автомата, принадлежащего классу У7, приведен на рисунке Б.5.

Введем в терминах автоматов класса У7 понятия, связанные с обработкой массивов.

Проход по массивам — часть вычисления автомата, в начальном состоянии которой текущей локацией является локация B, а в конечном — локация C; при этом в начальном состоянии этой части вычисления переменная i равна 0, в конечном — $n-1$, и в вычислении отсутствуют переходы с изменением значения переменной i с $n-1$ на 0.

При проходе по массивам переменная i последовательно принимает все значения от 0 до $n-1$, поскольку i изменяется только при выполнении функции $next()$ при переходе из локации

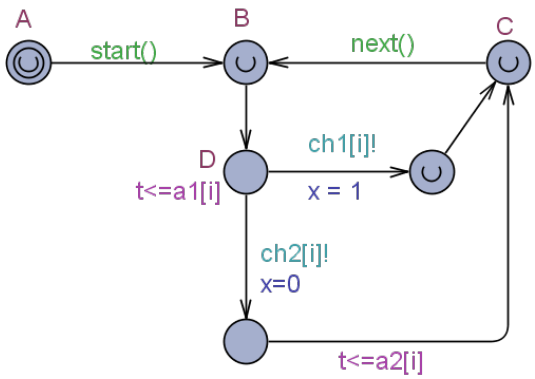

Рисунок Б.5 — Пример автомата, принадлежащего классу У7

C в локацию B; при каждом выполнении этой функции если i не равно $n-1$, то i увеличивается на 1, иначе i становится равным 0.

Итерация обработки массивов — часть вычисления автомата (последовательность дискретных и непрерывных переходов между состояниями), в начальном состоянии которой текущей локацией является локация B, а в конечном — локация C; при этом во всех остальных состояниях этой части вычисления локации B и C не являются текущими. Одна итерация обработки массивов соответствует обработке элементов массивов с одним номером.

В рамках итерации обработки массивов переменная i имеет фиксированное значение, поскольку она изменяется только при выполнении функции $next()$ при переходе из локации C в локацию B. По определению, итерация обработки массивов не содержит переходов из локации C в локацию B.

Проход по массивам состоит из n итераций обработки массивов, разделенных переходами из локации C в локацию B, на каждом из которых выполняется функция $next()$.

Лемма 2. Пусть для параметризованного автомата из класса У7, работающего с временными параметрами (в т.ч. массивами временных параметров), дополнительно выполнены следующие условия.

1. Используемые в автомате элементарные сравнения таймеров имеют вид $t\ op\ p^*$, либо $t\ op\ p[i]$, где p^* — временной параметр, не являющийся элементом массива, p — массив временных параметров ($p[i]$ — элемент массива временных параметров), op — одна из операций «<=», «==».
2. Инварианты всех локаций, в которых разрешено продвигать время, имеют вид конъюнкции произвольного логического выражения, не содержащего сравнений таймера t, и выражения одного из двух видов:
   а) $t <= p[i]$, где p — массив временных параметров;
   б) $(t <= p^*)$ && $(f == 1)$, где p^* — временной параметр, не являющийся элементом массива, f — признак обработки последних элементов массивов, см. п. 2 определения класса У7;
3. Для любых $j,h \in \overline{0,n-1}$, таких что $j < h$, для всех массивов временных параметров параметров $p\_1, ..., p\_q$, верно, что $max(p\_1[j], ..., p\_q[j]) \leqslant min(p\_1[h], ..., p\_q[h])$.

4. Для всех временных параметров $p^*\_1, ..., p^*\_r$, не являющихся элементами массивов $p\_1, ..., p\_q$, верно, что $max(p\_1[n-1], ..., p\_q[n-1]) \leqslant min(p^*\_1, ..., p^*\_r)$.

Тогда каждое элементарное сравнение вида $t <= p[i]$ или $t <= p^*$ имеет истинное значение во всех состояниях с текущей локацией B.

Доказательство.

Рассмотрим произвольное состояние s некоторого экземпляра параметризованного автомата из класса У7, которому (состоянию) соответствует локация B. Согласно условию 5 определения класса У7, состоянию s предшествовал переход с выполнением одной из функций $start()$ и $next()$.

Если в рассматриваемом состоянии s $i == 0$, то $t == 0$ (согласно виду функций $start()$ и $next()$) и все элементарные сравнения вида $t <= p[i]$ или $t <= p^*$ истинны, т.к. все временные параметры по определению имеют неотрицательные значения.

Рассмотрим случай, когда в рассматриваемом состоянии s $i == k$, $k > 0$. В этом случае состоянию s непосредственно предшествовал дискретный переход из локации C в локацию B с выполнением функции $next()$. Продвижение времени в локации B запрещено. Возможны два варианта:

- $t == 0$. Следовательно, все элементарные сравнения вида $t <= p[i]$ или $t <= p^*$ в s истинны, поскольку все временные параметры имеют неотрицательные значения. В этом варианте в текущем проходе по массивам (т.е. в проходе, которому принадлежит состояние s) нет непрерывных переходов до перехода в состояние s.
- $t > 0$. Значит, в текущем проходе по массивам имеются непрерывные переходы до перехода в состояние s. Рассмотрим конечное состояние s^* того непрерывного перехода из текущего прохода по массивам, который ближе всего к состоянию s. Значения таймера t в состояниях s и s^* совпадают. Обозначим за m значение переменной i в состоянии s^*. $m < k$, поскольку при переходе в состояние s значение переменной i увеличивается на 1 при выполнении функции $next()$. Инвариант локации, соответствующей состоянию s^*, является истинным в s^*. Этот инвариант не может иметь вид $(t <= p^*)$ && $(f == 1)$, поскольку, если бы это выражение было истинным в s^*, то в s^* переменная i имела бы значение $n-1$ (т.к., согласно виду функции $next$, f равно 1 тогда и только тогда, когда значение переменной i равно $n-1$) и, следовательно, в s переменная i имела бы значение 0 (согласно виду функции $next()$), что противоречит предположению $i == k$, $k > 0$. Поэтому инвариант локации, соответствующей состоянию s^*, имеет вид $t <= p[i]$. Следовательно, согласно условию 3 леммы, в состоянии s^* выполняется неравенство $t <= max(p\_1[m], ..., p\_q[m]) <= min(p\_1[k], ..., p\_q[k])$. С учетом выполнения этого неравенства, из того, что при $i == k$ в элементарных сравнениях вида $t <= p[i]$ или $t <= p^*$ могут использоваться только временные параметры $p\_1[k], ..., p\_q[k]$ и $p^*\_1, ..., p^*\_r$, из равенства значений таймера t в состояниях s^* и s, а также из условия 4 леммы, следует что в s все элементарные сравнения вида $t <= p[i]$ или $t <= p^*$ истинны.

Лемма доказана.

Лемма 3. Пусть параметризованный автомат принадлежит классу У7, и для него выполнены условия леммы 2. Тогда для любого экземпляра этого автомата и любой последовательности

синхронизаций, принадлежащей некоторому проходу по массивам в некотором вычислении этого экземпляра автомата, существует вычисление этого экземпляра автомата, в рамках которого первый проход по массиву содержит рассматриваемую последовательность синхронизаций.

Доказательство.

Пусть X — экземпляр автомата, принадлежащего классу У7 и удовлетворяющего условиям леммы. Рассмотрим вычисление S этого экземпляра автомата, включающее проход по массивам, в котором содержится рассматриваемая последовательность синхронизаций. Этот проход представляет собой последовательность переходов $s_1 \xmapsto{as_1, aa_1} s_2 \dots s_{n-1} \xmapsto{as_{n-1}, aa_{n-1}} s_n$. Построим на основе этой последовательности переходов вычисление S^*, первый проход по массивам в котором содержит рассматриваемую последовательность синхронизаций.

В качестве начального состояния s_1^* вычисления S^* возьмем начальное состояние s_1 вычисления S. Далее, согласно условию 5 определения класса У7, в вычислении S следует дискретный переход из локации A в локацию B с выполнением функции $start()$. Этот переход является единственным возможным переходом из начального состояния. Помещаем этот переход в S^*. Конечные состояния рассмотренного перехода в обоих вычислениях (s_2 и s_2^*) являются начальными состояниями первого прохода по массивам. Согласно виду функций $start()$ и $next()$, начальные состояния всех проходов по массивам, в т.ч. первого прохода, идентичны (их текущие локации, значения переменных и таймера совпадают).

Пусть s_k — начальное состояние того прохода по массивам в S, который содержит рассматриваемую последовательность синхронизаций. Это состояние идентично состоянию s_2.

Далее будем последовательно добавлять в частично построенное вычисление S^* все переходы из S, начиная с перехода из состояния s_k. Каждый такой переход возможен из конечного состояния последнего добавленного перехода в S^*, так как:

- этот переход имеет место в S из состояния, соответствующего конечному состоянию последнего добавленного перехода в S^*;
- все соответствующие друг другу состояния вычисления S и уже построенной части вычисления S^*, идентичны;
- рассматриваемый автомат не имеет переменных интерфейса и выполняется условие 7 определения класса У7; поэтому любой другой автомат, функционирующий совместно с экземпляром автомата X, не может повлиять на возможность какого бы то ни было перехода в экземпляре автомата X ни посредством переменных, ни посредством синхронизаций.

Описанным образом построим вычисление S^* экземпляра автомата X, содержащее один проход по массивам, который включает требуемую последовательность синхронизаций.

Лемма доказана.

Сформулируем и докажем утверждение 7 и следствие из него, определяющие условия, при которых при верификации автоматов достаточно рассматривать массивы длины 1 и 2.

В отличие от утверждений 3—5, а также приведенного далее утверждения 8, применять которые целесообразно к автомату, эквивалентному сети из автомата-модели и автомата-наблюдателя, утверждение 7 ориентировано на применение к автомату-модели, функционирующему *совместно* с автоматом-наблюдателем. А следствие из утверждения 7 — к

автомату-наблюдателю. Причиной этой особенности является то, что в утверждении 7 содержатся ограничения на *структуру* автомата, которая не сохраняется при преобразовании сети автоматов к одному автомату.

Утверждение 7. Пусть параметризованный автомат принадлежит классу У7, и для него выполнены условия леммы 2. Тогда для любой последовательности синхронизаций, имеющей место *на одной итерации* обработки массивов произвольного размера, верно следующее: эта последовательность синхронизаций с теми же моментами синхронизаций имеет место и на некоторой итерации обработки массивов размера 1 или 2.

Доказательство.

Докажем утверждение 7 методом от противного. Пусть X — экземпляр параметризованного автомата, удовлетворяющего всем условиям определения класса У7 и всем условиям леммы 2. Пусть для X размер массивов равен k ($k > 2$), и для X существует такая последовательность синхронизаций seq, что она имеет место на i-й итерации обработки массивов, т.е. при обработке i-х элементов массивов для некоторого $i < k$. Предположим, что эта последовательность синхронизаций не имеет место ни на какой итерации обработки массивов ни для каких экземпляров рассматриваемого параметризованного автомата, размер массивов в которых не превышает 2 (ни при каких значениях элементов этих массивов).

Рассмотрим вычисление S экземпляра X, содержащее один проход по массивам, который включает последовательность синхронизаций seq. Такое вычисление всегда существует согласно лемме 3.

Рассмотрим все возможные значения i, при которых в S может иметь место последовательность синхронизаций seq (одной итерации обработки массивов соответствует одно значение i):

1. $i == 0$. Рассмотрим экземпляр X' того же параметризованного автомата, отличающийся от X лишь тем, что размер массивов в нем равен 2. Значения первых элементов каждого массива для X' совпадают со значениями первых элементов соответствующего массива для X. Для экземпляра автомата X' имеет место вычисление S', в котором 0-я итерация обработки массивов совпадает с 0-й итерацией обработки массивов в S, поскольку начальные состояния вычислений S и S' совпадают и значения всех используемых в автомате выражений (в т.ч. в функции $start()$) при $i == 0$ совпадают (первый вызов функции $next()$ выполняется *после* завершения 0-й итерации обработки массивов). Таким образом рассматриваемая последовательность синхронизаций имеет место при обработке 0-х элементов массивов в некотором экземпляре автомата, а именно в экземпляре X, в котором размер массивов равен 2.

Замечание. В данном случае рассматриваются массивы длины 2, а не 1, т.к. если бы в X' размер массивов был равен 1, то значения выражения $i == n - 1$, использующегося в функции $start()$, в X и X' различались бы при $i == 0$, и гарантировать совпадение 0-х итераций обработки массивов в X и X' было бы нельзя.

2. $i == k - 1$. В этом случае сформируем новые массивы размера 2, в каждом из которых значение 0-го элемента совпадает со значением 0-го элемента в соответствующем массиве экземпляра X рассматриваемого параметризованного автомата, а значение 1-го элемента – со значением $(k-1)$-го (последнего) в соответствующем массиве X. Рассмотрим экземпляр X' того же параметризованного автомата, отличающийся от X лишь тем, что в нем используются сформированные

указанным образом массивы размера 2. Условия 3 и 4 леммы 2 накладывают ограничения на значения элементов массивов. Для сформированных массивов эти условия выполняются: условие 3 — т.к. для каждого массива *порядок следования* значений элементов этого массива в автоматах X и X' совпадает; условие 4 — т.к. для каждого массива значения последнего элемента в автоматах X и X' совпадают.

На основе вычисления S экземпляра автомата X сформируем вычисление S' экземпляра автомата X', работающего с новыми массивами, следующим образом:

- В качестве начального состояния S' возьмем начальное состояние S. Помещаем в S' переход из локации A в локацию B с выполнением функции $start()$ (этот переход является первым в вычислении S). Переменная f при выполнении этого перехода в обоих вычислениях принимает значение 0.
- Далее рассмотрим последовательно все переходы в S, начиная с перехода из локации B в локацию D и до перехода, при котором выполняется функция $next()$, в результате чего i меняет значение с 0 на 1 (переход из C в B). Будем последовательно помещать каждый такой переход в S'. При этом соответствующие состояния в S и S', кроме последнего (после перехода из C в B), будут идентичны, поэтому каждый из рассматриваемых переходов в S возможен и в S'. Сформированная таким образом часть вычисления S' будет соответствовать обработке 0-х элементов массивов и переходу к следующим (1-м) элементам. Т.к. в S' размер массивов равен 2, то переменная f при выполнении этого перехода примет значение 1 (в отличие от соответствующего перехода в S).
- Далее пропустим переходы в S до перехода из локации C в локацию B, при котором значение i изменяется с $k-2$ на $k-1$ в функции $next()$, включительно (f при выполнении этого перехода в S примет значение 1, так же как при выполнении последнего добавленного в S' перехода).
- Далее в вычислении S следует переход из локации B в локацию D. Этот переход безусловный. Инвариант локации D имеет вид $t <= p[i]$ (согласно условию 8 определения класса У7), поэтому, согласно лемме 2, в конечном состоянии последнего добавленного в S' перехода (перехода в локацию B) этот инвариант имеет истинное значение. Добавляем в S' переход из локации B в локацию D.

  В состояниях-источниках этого перехода в S и в S' значения всех переменных, кроме переменной i, совпадают. Также в этих состояниях совпадают значения каждого выражения вида $r[i]$ (по построению массивов для экземпляра автомата X'). Согласно условию 8 определения класса У7, при выполнении действий этого перехода не используется и не изменяется значение таймера. Поэтому и в конечных состояниях перехода в S и в S' значения всех переменных, кроме переменной i, совпадают (*).

  Переходим к рассмотрению следующего перехода в S.
- Состояние-источник текущего перехода в S имеет вид $(D, c, \overline{v})$, где c — значение таймера c, $\overline{v}$ — набор значений переменных. Инвариант локации D имеет вид $t <= p[i]$, поэтому $c <= p[i]$ в этом состоянии в S. Конечное состояние последнего добавленного перехода в S' имеет вид $(D, c', \overline{v'})$. $c' <= c$, т.к. c' — значение таймера t в конце 0-й итерации обработки массивов в S, а c — значение таймера t в конце $(k-2)$-й итерации обработки массивов

в S, и значение таймера t не обнуляется в рамках одного прохода по массивам (согласно виду функции $next()$). В конечном состоянии последнего добавленного перехода в S' и в состоянии-источнике текущего перехода в S значения выражения $p[i]$ совпадают. Поэтому в конечном состоянии последнего добавленного перехода в S' выражение $c <= p[i]$ истинно. Следовательно, из этого состояния в S' возможен непрерывный переход, при котором значение таймера t увеличивается до значения c. Добавляем такой непрерывный переход в S'. Текущий рассматриваемый переход в S при этом остается тем же.

Для последнего добавленного состояния в S' и состояния-источника текущего перехода в S верно следующее:

а) Текущие локации в этих состояниях совпадают (локация D).

б) Значения каждой переменной, кроме переменной i, совпадают (вывод (*) на стр. 186). При этом i имеет значение 1 в S' и $(k-1)$ — в S.

в) Значения таймера t совпадают и равны c.

Таким образом, рассматриваемые состояния в S и S' различаются только значениями переменной i (которая, согласно условию 6 определения класса У7, может использоваться вне функций $start()/stop()$ только в конструкции «$[i]$»). При этом значения каждого выражения вида $r[i]$ в рассматриваемых состояниях S и S' совпадают (по построению массивов для экземпляра автомата X'). Поэтому для рассматриваемого перехода в S в рассматриваемом состоянии в S' значения условия этого перехода и инварианта локации-назначения совпадают с соответствующими значениями в S и равны $True$ (т.к. совпадают значения всех использующихся в них элементарных сравнений). Кроме того, согласно условию 7 определения класса У7, любая синхронизация может быть выполнена в экземпляре автомата X'. Следовательно, рассматриваемый переход возможен и из последнего добавленного состояния в S'. Добавляем этот переход в S'. Этот переход принадлежит $(k-1)$-й итерации обработки массивов в S (обработке элементов с номером $k-1$) и 1-й итерации обработки массивов в S'.

Для двух состояний, следующих непосредственно за рассматриваемым переходом в S и последним добавленным переходов в S', верно следующее (♡):

а) Текущие локации в этих состояниях совпадают.

б) Значения каждой переменной, кроме переменной i, совпадают (следует из совпадения значений переменных до выполнения перехода и из совпадения значений выражений вида «$r[i]$»). При этом i имеет значение 1 в S' и $(k-1)$ — в S.

в) Значения таймера t совпадают (следует из совпадения значений таймера до выполнения перехода и из совпадения значений выражений вида «$r[i]$»).

Поэтому, с учетом выполнения условия 7 определения класса У7, и следующий переход из S возможен в S'. Он также принадлежит $(k-1)$-й итерации обработки массивов в S и после добавления в S' будет принадлежать 1-й итерации обработки массивов в S'.

– Продолжим рассматривать переходы из S и добавлять их последовательно в S' до тех пор, пока S' не будет содержать последовательность переходов, поэлементно соответствующую последовательности переходов из S, содержащих последовательность

синхронизаций seq. После каждого перехода выполняются свойства ♡, поэтому следующий за ним переход возможен в S'. Поскольку последовательность синхронизаций seq соответствует одной итерации обработки массивов в S (а именно $(k-1)$-й итерации), то и в S' эта последовательность синхронизаций будет соответствовать одной итерации обработки массивов (а именно 1-й итерации).

Таким образом и S' будет содержать последовательность синхронизаций seq. Моменты синхронизаций в S' будут совпадать с моментами соответствующих синхронизаций в S, исходя из вида используемых в автоматах элементарных сравнений и совпадения значений выражений вида «$r[i]$» в соответствующих состояниях вычислений S и S'.

Таким образом, получим вычисление S' экземпляра X', содержащее обработку двух элементов массивов и рассматриваемую последовательность синхронизаций, что противоречит введенному предположению о том, это эта последовательность синхронизаций невозможна для любых массивов размера 2.

3. $i == u, 1 <= u < k-1$. В этом случае сформируем новые массивы размера 2, в каждом из которых значение 0-го элемента совпадает со значением u-го элемента в соответствующем массиве экземпляра X рассматриваемого параметризованного автомата, а значение 1-го элемента – со значением $(u+1)$-го в соответствующем массиве X. Рассмотрим экземпляр X' того же параметризованного автомата, отличающийся от X лишь тем, что в нем используются сформированные указанным образом массивы размера 2. Аналогично случаю $i == k-1$ для сформированных массивов выполняются условия 3 и 4 леммы 2.

На основе вычисления S автомата X сформируем вычисление S' автомата X' следующим образом:

- В качестве начального состояния S' возьмем начальное состояние S. Помещаем в S' переход из локации A в локацию B с выполнением функции $start()$ (этот переход является первым в вычислении S). Переменная f при выполнении этого перехода в обоих вычислениях принимает значение 0.
- Пропускаем все переходы в S до перехода из локации C в локацию B, при котором переменная i в результате выполнения функции $next()$ принимает значение u, включительно. Конечное состояние этого перехода в S отличается от последнего имеющегося на данный момент состояния S' только значениями переменной i и таймера t.
- Добавляем в S' переход из локации B в локацию D (такой же переход имеется в S).
- Рассматриваем значение таймера в конечном состоянии перехода из локации B в локацию D в вычислении S при $i == u$. Добавляем в S' непрерывный переход с продвижением времени до этого значения.
- Последовательно рассматриваем и добавляем в S' все переходы из S, начиная с перехода, следующего за переходом в локацию D при $i == u$. Возможность этих переходов обосновывается аналогично п. 2. Действия этого шага выполняем до тех пор, пока S' не будет содержать последовательность переходов, поэлементно соответствующую последовательности переходов из S, содержащих последовательность синхронизаций seq. Аналогично п. 2, последовательность синхронизаций seq будет соответствовать одной итерации обработки массивов в S' (а именно 0-й итерации). Также по аналогии с п. 2,

моменты синхронизаций в S' будут совпадать с моментами соответствующих синхронизаций в S.

Таким образом, аналогично п. 2, получим вычисление S', содержащее обработку двух элементов массивов и рассматриваемую последовательность синхронизаций, что противоречит введенному предположению.

Все возможные значения i рассмотрены, в каждом из случаев получено противоречие. Утверждение 7 доказано.

Следствие. Пусть параметризованный автомат удовлетворяет всем условиям определения класса У7, кроме условий 6 и 7. Пусть также этот автомат удовлетворяет условиям леммы 2 и является автоматом-наблюдателем, построенным согласно определению (см. раздел 3.2.1). Вместо условия 6 определения класса У7 этот автомат должен удовлетворять следующему условию 6':

6'. Единственными допустимыми конструкциями, содержащими оператор «[]», являются конструкции следующих видов:

- «$r[i]$», где r — используемый в автомате массив временных параметров или каналов, i — единственная индексная переменная автомата (в т.ч. допустимы конструкции $r[i]!$ и $r[i]?$).
- Разметка перехода вида «$select\ \ j : [0,n-1]; j\ != i; ch[j]!;$» либо «$select\ \ j : [0,n-1]; j\ != i; ch[j]?;$», где ch — используемый в автомате массив каналов. При этом локацией назначения является «плохая» локация автомата-наблюдателя.

Вне функций $start()$ и $next()$, в т.ч. в других функциях автомата, переменная i может использоваться только в конструкциях приведенного выше вида.

Тогда при функционировании этого автомата совместно с автоматом, удовлетворяющим условиям утверждения 7, для «плохой» локации этого автомата-наблюдателя верно, что если она не достижима для любых значений элементов массивов размера 1, 2 и 3, то эта локация не достижима для любых значений элементов массивов любого размера.

Замечание. Перечисленных в условии 6' конструкций достаточно для того, чтобы обеспечить наличие активных переходов с действиями синхронизации, парными ко всем возможным действиям синхронизации автомата, функционирующего совместно с рассматриваемым автоматом-наблюдателем.

Доказательство.

В доказательстве используется понятие синхронности проходов по массивам. Согласно условию 5 определения класса У7 при переходе к следующей итерации обработки массивов в обоих автоматах «сбрасываются» значения всех переменных, кроме единственной индексной переменной. Таким образом, обеспечивается отсутствие влияния переходов одной итерации на переходы другой итерации. Однако, если номера обрабатываемых элементов в рассматриваемой паре автоматов не совпадают (например, в автомате-наблюдателе началась обработка i-х элементов массивов, а в автомате-модели еще выполняется обработка $(i-1)$-х элементов), то переходы $(i-1)$-й итерации обработки массивов в автомате-модели могут влиять (посредством синхронизаций и никак иначе) на переходы i-й итерации обработки массивов в автомате-наблюдателе. Для того, чтобы исключить данный эффект, в автомате-наблюдателе имеются переходы в «плохую» локацию с конструкцией $select$ вида, описанного в условии 6'. Согласно условию 6 определения класса У7, на i-й итерации обработки массивов в автомате-модели выполняются синхронизации

только по каналам-элементам массивов с номерами i. Согласно условию 6', если i-й итерации обработки массивов в автомате-наблюдателе происходит синхронизация по каналу с номером j, не равным значению индексной переменной i этого автомата, то это означает, что в автомате-модели обрабатываются элементы с номерами j, т.е. обработка массивов в рассматриваемой паре автоматов выполняется не синхронно. Поэтому последовательность синхронизаций, содержащая такую синхронизацию, считается некорректной, и при выполнении этой синхронизации выполняется переход в «плохую» локацию.

В рамках доказательства настоящего следствия будем считать, что для пары рассматриваемых параметризованных автоматов проходы по массивам выполняются синхронно, если во всех вычислениях автомата-наблюдателя отсутствуют переходы с синхронизациями по каналам с номерами j, где $j! = i$ в начальном состоянии перехода, то есть на момент синхронизаций значения индексных переменных в автоматах совпадают. При этом, поскольку автомат-модель взаимодействует с автоматам-наблюдателям только посредством синхронизаций, то совпадение значений индексных переменных *во всех* состояниях сети из этих двух автоматов не требуется.

Доказательство данного следствия выполняется в два шага. Сначала (шаг 1) доказывается, что если выполнены все условия следствия и «плохая» локация автомата-наблюдателя не достижима для любых значений элементов массивов размера 1, 2 или 3, то проходы по массивам выполняются синхронно для любых значений элементов массивов любого размера. Затем (шаг 2) доказывается, что если выполнены все условия следствия, проходы по массивам выполняются синхронно для любых значений элементов массивов любого размера, и «плохая» локация автомата-наблюдателя не достижима для любых значений элементов массивов размера 1 или 2, то эта локация не достижима для любых значений элементов массивов любого размера. Доказательства каждого шага выполняются методом от противного.

Шаг 1. Пусть X_N и X_M — экземпляры автомата-наблюдателя и автомата-модели, удовлетворяющих всем условиям настоящего следствия. Пусть для них размер массивов равен k ($k > 3$) и в некотором вычислении S_N экземпляра X_N имеется переход с синхронизацией по каналу с номером j, где $j! = i$ в начальном состоянии этого перехода. Согласно условию 6' локацией назначения такого перехода является «плохая» локация. Из того, что а) каждая конкретная последовательность синхронизаций формируется исключительно за счет переходов в автомате-модели (из любого состояния автомата-наблюдателя возможен переход с синхронизацией по любому каналу), б) переход в «плохую» локацию в автомате-наблюдателе соответствует некорректной последовательности синхронизаций в автомате-модели, и в) в автомате-модели отсутствует «память» между итерациями, следует, что либо при обработке i-го элемента в автомате-модели отсутствует необходимая (ожидаемая в автомате-наблюдателе) синхронизация по i-му каналу (в результате в автомате-наблюдателе не выполняется переход к j-й итерации), либо в автомате-модели присутствует «лишняя» синхронизация по j-му каналу, когда в автомате-наблюдателе ожидается синхронизация по i-му каналу. В обоих случаях наличие рассматриваемого перехода говорит о некорректной последовательности синхронизаций при обработке i-х или j-х элементов массивов автоматом-моделью. Рассматриваемому вычислению S_N экземпляра X_N соответствует вычисление S_M экземпляра X_M, i-я или j-я итерация которого содержит некорректную последовательность синхронизаций.

Пусть для определенности $i < j$ (для случая $i > j$ рассуждения аналогичны). Возможны три варианта.

1. $j! = k - 1$. В этом случае сформируем новые массивы размера 3, в каждом из которых значение 0-го элемента совпадает со значением i-го элемента в соответствующем массиве экземпляра X_M, значение 1-го элемента — со значением j-го элемента в соответствующем массиве X_M, значение 2-го элемента — со значением $(k-1)$-го (последнего) в соответствующем массиве X_M. Рассмотрим экземпляры X'_M и X'_N тех же параметризованных автоматов, отличающиеся от X_M и X_N лишь тем, что в них используются сформированные указанным образом массивы размера 3. На основе вычисления S_M экземпляра X_M строим вычисление S'_M, содержащее последовательности синхронизаций, соответствующие в S_M i-й и j-й итерациям обработки массивов. Способ построения вычисления S'_M по вычислению S_M совпадает со способом построения вычисления S' по вычислению S в доказательстве утверждения 7.

В результате получим вычисление S'_M экземпляра X'_M, работающего с массивами размера 3, которому (вычислению) соответствует вычисление S'_N экземпляра X'_N, работающего с массивами размера 3, содержащее переход в «плохую» локацию (по построению автомата-наблюдателя, распознающего все некорректные последовательности синхронизаций, в т.ч. синхронизации по каналам $ch[j]$ при $i! = j$). Тем самым получим противоречие.

2. $j == k-1$, $i == 0$. В этом случае сформируем новые массивы размера 3, в каждом из которых значение 0-го элемента совпадает со значением i-го (т.е. 0-го) элемента в соответствующем массиве экземпляра X_M, значение 1-го элемента — со значением 1-го элемента в соответствующем массиве X_M, значение 2-го элемента — со значением j-го (т.е.$(k-1)$-го, последнего) в соответствующем массиве X_M. Рассмотрим экземпляры X'_M и X'_N тех же параметризованных автоматов, отличающиеся от X_M и X_N лишь тем, что в них используются сформированные указанным образом массивы размера 3. Аналогично п. 1, получим вычисление экземпляра X'_N, содержащее переход в «плохую» локацию и тем самым получим противоречие.

3. $j == k-1$, $i! = 0$. В этом случае сформируем новые массивы размера 3, в каждом из которых значение 0-го элемента совпадает со значением 0-го элемента в соответствующем массиве экземпляра X_M, значение 1-го элемента — со значением i-го элемента в соответствующем массиве X_M, значение 2-го элемента — со значением j-го (т.е.$(k-1)$-го, последнего) в соответствующем массиве X_M. Рассмотрим экземпляры X'_M и X'_N тех же параметризованных автоматов, отличающиеся от X_M и X_N лишь тем, что в них используются сформированные указанным образом массивы размера 3. Аналогично п. 1, получим вычисление экземпляра X'_N, содержащее переход в «плохую» локацию и тем самым получим противоречие.

Во всех трех случаях формирование новых массивов выполнялось так, чтобы для них выполнялись ограничения леммы 2, необходимые для построения новых вычислений, аналогично такому построению в доказательстве утверждения 7.

Итого, во всех трех случаях получено противоречие и, следовательно, введенное предположение неверно, и синхронность проходов по массивам доказана.

Шаг 2.

Докажем, что если выполнены все условия следствия, проходы по массивам выполняются синхронно для любых значений элементов массивов любого размера, и «плохая» локация

автомата-наблюдателя не достижима для любых значений элементов массивов размера 1 или 2, то эта локация не достижима для любых значений элементов массивов любого размера.

Доказательство этого факта аналогично доказательству утверждения 7 со следующими отличиями:

– Рассматривается вычисление S некоторого экземпляра автомата-наблюдателя, содержащее не заданную последовательность синхронизаций (как в случае автомата-модели), а переход в «плохую» локацию при работе автомата-наблюдателя с массивами длины $k > 2$. На основе этого вычисления формируются массивы размера 2, и для нового экземпляра автомата, отличающегося от исходного лишь тем, что он работает со сформированными массивами размера 2, строится вычисление S', содержащее переход в «плохую» локацию.
– В доказательстве утверждения 7 используется тот факт, что в автомате-модели значения всех переменных, кроме индексной переменной i (которая «изолирована» от остальных переменных), «сбрасываются» в начале каждой итерации обработки массивов, а во всех локациях автомата-наблюдателя активны переходы с действиями синхронизации, парными ко всем возможным действиям синхронизации исходного автомата. Таким образом, в автомате-модели отсутствует «память» между итерациями, а автомат-наблюдатель не может «повлиять» на последовательность переходов автомата-модели, даже если у него такая «память» есть. В отличие от этого, при доказательстве следствия из утверждения 7 используется факт отсутствия «памяти» между итерациями в автомате-наблюдателе, факт отсутствия «памяти» между итерациями в автомате-модели, а также синхронность проходов по массивам автомата-модели и автомата-наблюдателя, доказанная на шаге 1. Поэтому в обоих случаях для экземпляра автомата-модели (автомата-наблюдателя), работающего с массивами произвольной длины, при функционировании которого на i-й итерации имеет место некоторая последовательность синхронизаций (достижима «плохая» локация), можно построить экземпляр этого автомата, работающий с массивами длины 2, при функционировании которого на 0-й или 1-й итерации имеет место эта последовательность синхронизаций (достижима «плохая» локация).

Итого, по совокупности двух шагов доказательства доказано, что если выполнены все условия следствия и «плохая» локация не достижима для любых значений элементов массивов размера 1, 2 и 3, то эта локация не достижима для любых значений элементов массивов любого размера. Следствие доказано.

Замечание. Если для построения некоторой модели требуется использовать как конструкции, описанные в утверждении 6, так и конструкции, описанные в утверждении 7, то рекомендуется представлять такую модель как композицию нескольких автоматов, каждый их которых в отдельности удовлетворяет одному из приведенных утверждений, и верифицировать построенные автоматы отдельно.

Таким образом, если автомат-модель и автомат-наблюдатель удовлетворяют соответствующим условиям следствия из утверждения 7, то при проверке достижимости «плохой» локации в автомате-наблюдателе достаточно проверить ее достижимость для массивов размеров 1, 2 и 3. Элементами массивов в данном случае являются каналы и временные параметры. Вопрос выбора *значений* временных параметров-элементов массивов рассмотрен в разделе Б.2.

В данной работе следствие из утверждения 7 применяется для проверки выполнения требований к модели планировщика ядра.

Б.4. Индексные параметры

Для доступа к элементам массивов, помимо индексных переменных, используются *индексные параметры*. Это целые неотрицательные параметры, которые могут использоваться для обращения к элементу массива по индексу с помощью оператора []. В данной работе индексные параметры используются только для обращения к элементам массивов каналов. Массивы индексных параметров используются для обращения к элементам массивов каналов, в т.ч. разные элементы массивов индексных параметров могут использоваться для обращения к одному и тому же элементу массива каналов. В частности, массивы индексных параметров используются моделью планировщика ядра. Для каждого окна в расписании имеется индексный параметр-элемент массива, задающий номер раздела, которому принадлежит это окно. Этот параметр определяет номер каналов в массивах каналов $sleep[]$ и $wakeup[]$, по которым выполняются синхронизации, моделирующие открытие и закрытие рассматриваемого окна.

Отметим, что если имеется набор каналов и массив каналов c (например, массив, рассматриваемый в утверждении 7 и следствии из него), элементами которого являются элементы указанного набора (в т.ч. один элемент набора может входить в массив несколько раз), то этот массив каналов может быть заменен на пару массивов: массив уникальных каналов c' и массив индексных параметров p', такие, что размеры массивов c и p' совпадают, и для любого i верно, что $c[i] == c'[p'[i]]$.

Далее будет приведено утверждение 8, позволяющее сократить для индексных параметров диапазон значений, перебираемых при верификации. Для доказательства этого утверждения используется вспомогательная лемма 4.

Лемма 4. Пусть параметризованный автомат имеет k целых неотрицательных параметров p_1, ..., p_k, использующихся *только* в условиях переходов; причем эти условия имеют вид $p\ != p^*$, где параметры p, p^* принадлежат набору p_1, ..., p_k. Тогда для любой локации этого автомата верно, что если она недостижима при любых целых неотрицательных значениях параметров p_1, ..., p_k, не превышающих $k - 1$, то она недостижима при любых целых неотрицательных значениях этих параметров.

Доказательство.

Докажем лемму 4 методом от противного. Предположим, что для некоторого параметризованного автомата, удовлетворяющего условиям леммы 4, некоторая локация E не достижима при любых целых неотрицательных значениях параметров p_1, ..., p_k, не превышающих $k - 1$, но достижима при некоторых значениях параметров $p\_1 = m_1$, ..., $p\_k = m_k$, таких что хотя бы одно из значений m_1, ..., m_k превосходит $k - 1$.

Рассмотрим экземпляр X этого параметризованного автомата, в котором параметры p_1, ..., p_k имеют значения m_1, ..., m_k. Согласно предположению, этот экземпляр автомата имеет вычисление S_x, содержащее переход в локацию E.

Удалим из набора m_1, ..., m_k дубликаты и тем самым получим набор различных целых неотрицательных чисел a_1, ..., a_l, $l \leqslant k$. Сформируем набор целых неотрицательных чисел n_1, ..., n_k следующим образом: $\forall i \in \overline{1,k}\ n_i = h - 1$, где h такое, что $m_i = a_h$, т.е. h — номер числа m_i в наборе a_1, ..., a_l. Для сформированного набора верно, что $\forall i,j \in \overline{1,k}$, если $m_i = m_j$, то и $n_i = n_j$ (поскольку если $a_h = a_g$, то $h - 1 = g - 1$). Аналогично, $\forall i,j \in \overline{1,k}$, если $m_i \neq m_j$, то и $n_i \neq n_j$. При этом $\forall i \in \overline{1,k}\ n_i \leqslant k - 1$.

Рассмотрим экземпляр Y того же параметризованного автомата, отличающийся от экземпляра X лишь тем, что в нем параметры p_1, ..., p_k имеют значения n_1, ..., n_k. Для автомата Y существует вычисление S_y, аналогичное вычислению S_x, и отличающееся от S_x только *видом* условий переходов, в которых используются значения параметров: для S_x — «$m_i\ !\ = m_j$», для S_y — «$n_i\ !\ = n_j$». При этом *логические значения* условий в вычислениях S_x и S_y совпадают. Таким образом, локация E достижима для набора n_1, ..., n_k значений параметров p_1, ..., p_k, не превышающих $k - 1$. Следовательно, введенное предположение неверно и лемма 4 доказана.

Утверждение 8. Пусть задан параметризованный автомат M, удовлетворяющий следующим условиям:

1. автомат M имеет k индексных параметров p_1, ..., p_k;
2. интерфейс автомата M включает синхронизации по каналам «точка-точка», являющимся элементами массивов каналов c_1, ..., c_l; l — количество массивов каналов; размеры всех массивов каналов одинаковы и задаются параметром q;
3. для параметров верны соотношения $p\_1 < q$, ..., $p\_k < q$;
4. параметры p_1, ..., p_k и массивы каналов c_1, ..., c_l используются *только* в разметках переходов вида «; ; $c[p]$!;» и «; ; $c[p]$?;», где c — один из массивов каналов c_1, ..., c_l; p — один из параметров p_1, ..., p_k;
5. параметр q не используется ни в каких выражениях автомата M.

Пусть также задан параметризованный автомат N, удовлетворяющий следующим условиям:

1. автомат N работает с индексными параметрами p_1, ..., p_k автомата M;
2. интерфейс автомата N включает синхронизации по каналам «точка-точка», являющимся элементами массивов каналов c_1, ..., c_l; при этом интерфейс автомата N комплиментарен интерфейсу автомата M: если в автомате M выполняется отправка сигналов по некоторому каналу, то в автомате N выполняется прием сигналов по этому каналу, и наоборот;
3. в автомате N параметры p_1, ..., p_k и массивы каналов c_1, ..., c_l используются *только* в разметках переходов вида «; ; $c[p]$!;», «; ; $c[p]$?;», «$select\ i : [0, q–1]; i!=p; c[i]!$;», «$select\ i : [0, q–1]; i!= p; c[i]?$;», «$select\ i : [0, q–1]; ; c[i]!$;», «$select\ i : [0, q–1]; ; c[i]?$;», где c — один из массивов каналов c_1, ..., c_l, p — один из параметров p_1, ..., p_k;
4. в автомате N параметр q используется только в конструкциях указанного в п. 3 вида.

Автоматы M и N функционируют совместно в сети из этих двух автоматов. Для каждого из автоматов M и N и каждого из каналов верно, что по заданному каналу в заданном автомате возможна либо только отправка сигналов, либо только прием.

Тогда для любой локации автоматов M и N верно, что если она недостижима при любых целых неотрицательных значениях параметров p_1, ..., p_k, не превышающих $k-1$, и при $q = k$, то она недостижима при любых целых неотрицательных значениях p_1, ..., p_k и любых натуральных значениях q, таких что $p\_1 < q$, ..., $p\_k < q$.

Доказательство.

Рассмотрим автомат MN, эквивалентный сети автоматов, состоящей из автоматов, описанных в условии. Он строится согласно правилам, приведенным на странице 140. Согласно этим правилам, каждой локации каждого из исходных автоматов M и N соответствует набор локаций автомата MN, и проверка достижимости некоторой локации исходного автомата (M или N) сводится к проверки достижимости хотя бы одной из локаций автомата MN, соответствующих этой локации.

Для краткости рассмотрим случай, когда интерфейс автомата M включает только отправку сигналов по каналам (следовательно, интерфейс автомата N включает только прием сигналов по каналам). В общем случае интерфейс автомата M может включать отправку сигналов по одним каналам и прием — по другим; рассуждения для этого случая будут такими же, как приведенные ниже.

По построению в автомате MN отсутствуют переходы с выполнением действий синхронизации. Поскольку в автоматах M и N в действиях по изменению переменных и обнулению таймеров, а также в инвариантах локаций, параметры p_1, ..., p_k не используются, то и в автомате MN в действиях по изменению переменных и обнулению таймеров, а также в инвариантах локаций, эти параметры не используются. Таким образом, в автомате MN параметры p_1, ..., p_k могут использоваться *только* в условиях переходов. Рассмотрим все переходы в автомате MN, в условиях которых используются эти параметры.

По построению автомата MN каждый переход в этом автомате соответствует либо переходу автомата M, не имеющему действия синхронизации; либо переходу автомата N, не имеющему действия синхронизации; либо паре переходов с комплиментарными действиями синхронизации (один — из автомата M, другой — из автомата N). Поскольку в автоматах M и N индексные параметры используются только в разметке переходов с действиями синхронизации, то в автомате MN индексные параметры могут использоваться только в условиях переходов, каждый из которых соответствует паре переходов автоматов M и N (по правилам построения MN, согласно которым синхронизации в MNотсутствуют). Рассмотрим все возможные такие пары переходов.

Рассмотрим каждый переход автомата M с действием синхронизации вида $c[p]!$, где c — один из массивов каналов c_1, ..., c_l; p — один из параметров p_1, ..., p_k. Согласно ограничению 4 на вид автомата M, этот переход имеет разметку вида «; ; $c[p]!$;». Если в автомате MN имеются переходы, соответствующие рассматриваемому переходу в автомате M, то в автомате N имеется не менее одного перехода, каждый из которых имеет один из описанных ниже видов (1, 2, 3). Рассмотрим каждый из этих переходов автомата N:

1. Переход с разметкой вида «; ; $c[p]?$;». В таком случае, согласно правилам со страницы 140, паре из этого перехода в автомате N и рассматриваемого перехода в автомате M соответствует один переход в автомате MN, в разметке которого не используются параметры p_1, ..., p_k и q.
2. Переход с разметкой вида «$select\ i : [0, q-1]$; ; $c[i]?$;». Это означает, что в N этой разметке соответствуют q переходов с действиями синхронизации $c[0]?$, ..., $c[q-1]?$. Поскольку $p < q$, то при любых p и q ровно один из перечисленных переходов в автомате N имеет комплиментарное к $c[p]!$ действие синхронизации. Паре из этого перехода в автомате N и рассматриваемого перехода в автомате M соответствует один переход в автомате MN, в разметке которого не используются параметры p_1, ..., p_k и q.
3. Переход с разметкой вида «$select\ i : [0, q-1]; i!{=}p^*; c[i]?$». Это означает, что в N этой разметке соответствуют q переходов с условиями $0! = p^*$, ..., $(q-1)! = p^*$ и действиями синхронизации $c[0]?$, ..., $c[q-1]?$. Аналогично предыдущему случаю, только один из перечисленных переходов в автомате N имеет комплиментарное к $c[p]!$ действие синхронизации. Это переход с условием $p!{=}p^*$ и действием синхронизации $c[p]?$. Паре из этого перехода в автомате N и рассматриваемого перехода в автомате M соответствует один переход в автомате MN с условием перехода $p!{=}p^*$, где p и p^* — параметры из набора p_1, ..., p_k.

Итого, в автомате MN параметры p_1, ..., p_k, используются только в условиях переходов вида $p!{=}p^*$, где параметры p, p^* принадлежат набору p_1, ..., p_k. Следовательно, автомат MN удовлетворяет условию леммы 4, и достижимость его локаций достаточно проверять при целых неотрицательных значениях параметров p_1, ..., p_k, не превышающих $k-1$.

Согласно ограничению 5 на вид автомата M, ограничению 4 на вид автомата N, и правилам построения автомата MN, параметр q (размер массивов каналов) не используется в автомате MN, но для того, чтобы в автомате MN присутствовал переход, соответствующий паре переходов с комплиментарными действиями синхронизации по каналу $c[p]$, необходимо, чтобы значение q в паре автоматов M, N было больше значения p для любого параметра p из набора p_1, ..., p_k. Поэтому при переборе значений параметров p_1, ..., p_k, не превышающих $k-1$, для параметра q можно использовать значение, равное k. Утверждение 8 доказано.

Б.5. Переменные интерфейса

В интерфейс верифицируемого автомата могут входить переменные, изменяемые другими автоматами, но не им самим, и требования к верифицируемому автомату должны выполняться при всех возможных значениях таких переменных, с учетом того, что значение каждой переменной интерфейса может поменяться на любом шаге вычисления автомата. В общем случае, при верификации необходимо рассматривать все возможные значения переменных интерфейса. Однако в некоторых случаях этого можно избежать. Так для планировщика, работающего по схеме EDF, не важно числовое значение правой границы директивного интервала готовой работы, а важен

лишь факт наличия других готовых работ с меньшими значениями правых границ директивных интервалов. В данной работе рассматриваются только такие автоматы, в которых целочисленные переменные интерфейса, изменяемые другими автоматами, могут сравниваться лишь друг с другом и не используются в операторе присваивания. Тем самым такие переменные изолированы от значений констант, параметров и внутренних переменных автоматов. Для булевых переменных интерфейса таких ограничений нет (для таких переменных нет потребности сокращения диапазона перебираемых значений, поэтому рассуждения настоящего раздела к булевым переменным не применяются).

Пусть имеется некоторое вычисление $s_1 \xrightarrow{as_1,aa_1} s_2 \ldots s_{i-1} \xrightarrow{as_{i-1},aa_{i-1}} s_i \xrightarrow{as_i,aa_i} \ldots$ автомата, работающего с переменными (в общем случае как с переменными интерфейса, так и с внутренними переменными). Для каждой переменной этому вычислению однозначно соответствует последовательность значений данной переменной. 1-й элемент этой последовательности — значение переменной в начальном состоянии автомата, i-й элемент — значение переменной в i-м состоянии вычисления. Для набора переменных рассматриваемому вычислению однозначно соответствует набор последовательностей значений этих переменных.

Утверждение 9. Пусть параметризованный автомат имеет набор переменных, среди которых имеется k целочисленных переменных (обозначим этот набор из k переменных символом I), удовлетворяющих следующим условиям:

1. Переменные I не могут использоваться в правой части операторов присваивания.
2. Из любой локации автомата имеется безусловный переход в нее же, при котором выполняется произвольное изменение переменных I. На других переходах автомата изменения этих переменных невозможны. Начальные значения переменных I устанавливаются произвольным образом.
3. Переменные I могут использоваться лишь в операциях сравнения вида $x_i \ op \ x_j$, где op — одна из операций $\{< , > , == , != , >=, <=\}$, x_i и x_j — переменные, принадлежащие набору I.

Тогда для любой локации рассматриваемого автомата верно, что если не существует вычисления, в котором достижима эта локация и для которого все значения в наборе последовательностей значений переменных I не превышают k, то не существует никакого вычисления, в котором достижима эта локация.

Замечание. Аналогично автомату, рассматриваемому в утверждении 6, автомат, удовлетворяющий условию 3, получается при преобразовании в один автомат сети автоматов, состоящей из автомата-модели компонента МВС и автомата-наблюдателя. Указанная группа переменных интерфейса соответствует переменным интерфейса автомата-модели компонента МВС, остальные переменные — внутренние переменные автоматов.

Доказательство.

Докажем утверждение 9 методом от противного. Предположим, что для некоторого автомата, удовлетворяющего условиям утверждения 9, некоторая локация E не достижима для любого вычисления, для которого все значения в наборе последовательностей значений переменных I не

превышают k, но достижима для некоторого вычисления S, для которого хотя бы одной из переменных I соответствует последовательность значений этой переменной, содержащая некоторое число, большее k.

По вычислению S будем строить другое вычисление S' этого же автомата, содержащее переход в локацию E, такое что все соответствующие вычислению S' последовательности значений переменных набора I не содержат чисел, больших k.

Рассмотрим начальное состояние вычисления S. Если в этом состоянии значения всех переменных I не превышают k, то возьмем это состояние в качестве начального состояния вычисления S'.

Если в начальном состоянии вычисления S значение хотя бы одной из переменных I превышает k, то сформируем начальные значения переменных I следующим образом. Пусть N_1 — набор значений переменных I в начальном состоянии вычисления S. Получим упорядоченный по возрастанию набор различных чисел $a_1, ..., a_l$, удалив дубликаты из набора N_1 и отсортировав оставшиеся значения по возрастанию. Так как количество переменных I равно k, то $l \leqslant k$. Сформируем набор N_2 значений переменных I следующим образом: если в наборе N_1 некоторая переменная имела значение a_i, то в наборе N_2 эта переменная имеет значение i. Таким образом, все значения в наборе N_2 не превышают k.

В качестве начального состояния вычисления S' возьмем состояние, отличающееся от начального состояния вычисления S лишь значениями переменных I: в начальном состоянии вычисления S набором значений переменных I является набор N_1, а в начальном состоянии вычисления S' — набор N_2. По построению набора N_2 верно, что для любого логического выражения вида $x_i \ op \ x_j$, где op — одна из операций $\{<\ ,\ >\ ,\ ==\ ,\ !=\ ,\ >=, <=\}$, x_i и x_j — переменные, принадлежащие набору I, значения этого выражения на наборах N_1 и N_2 совпадают. Так как переменные I могут использоваться только в выражениях указанного вида, в т.ч. не могут использоваться в операторе присваивания, и начальные состояния вычислений S и S' отличаются только значениями I, то для каждого используемого в автомате (в условиях и действиях переходов, а также в инвариантах локаций) логического выражения, содержащего в т.ч. таймеры, константы, параметры и переменные (как из I, так и другие), значение этого выражения в начальном состоянии S совпадает с его значением в начальном состоянии S'. Поэтому в начальном состоянии S' значение инварианта соответствующей локации совпадает со значением инварианта этой локации в начальном состоянии S и равно $True$. Также для любого перехода из начального состояния значения условия перехода будут совпадать для вычислений S и S'.

Начальные состояния вычислений S и S' отличаются только значениями переменных I; переменные I «изолированы» от других переменных, параметров и констант (согласно условию 3 утверждения 9) и не изменяются на переходах вида, отличного от описанного в условии 2 утверждения 9; значения каждого логического выражения вида $x_i \ op \ x_j$ (x_i и x_j — переменные из I) в начальных состояниях S и S' совпадают. Поэтому в вычислениях S и S' в результате выполнения действий на любом переходе из начального состояния, имеющем вид, отличный от описанного в условии 2, полученные состояния в вычислениях S и S' будут различаться только значениями переменных I. Следовательно, значения каждого используемого в автомате логического выражения, содержащего в т.ч. таймеры, константы, параметры и переменные (как из I, так и другие),

в полученных состояниях этих двух вычислений будут совпадать. Таким образом, для любого перехода из начального состояния, который имеет вид, отличный от описанного в условии 2, значения инварианта локации назначения (после выполнения действий перехода) будут совпадать для вычислений S и S'.

Поэтому любой переход, соответствующий возможному переходу из начального состояния S, возможен из начального состояния S' (переход вида, описанного в условии 2, является безусловным и возможен всегда). Переходы в S', соответствующие переходам в S, вводятся по аналогии с п. 2 (для дискретных переходов) и п. 3 (для непрерывных переходов) доказательства утверждения 6 (см. стр. 174 и стр. 176 соответственно).

Для пошагового построения S' по S имеют место перечисленные ниже факты, выполняющиеся по построению для начального состояния S', а для последующих состояний следующие из способа построения S' (★):

1. В текущем состоянии вычисления S', т.е. в конечном состоянии уже построенной части S', значения переменных I не превышают k.
2. Текущее состояние вычисления S' и начальное состояние рассматриваемого перехода в S различаются только значениями переменных I.
3. Значение каждого использующегося в автомате логического выражения, содержащего в т.ч. таймеры, константы, параметры и переменные (как из I, так и другие), в предшествующем рассматриваемому переходу состоянии вычисления S совпадает со значением этого логического выражения в текущем состоянии вычисления S'. Это значит, что из истинности условия рассматриваемого перехода в предшествующем этому переходу состоянии S следует истинность условия соответствующего перехода в текущем состоянии вычисления S'.

Будем строить вычисление S' по S, обосновывая при этом выполнение фактов ★. Рассмотрим последовательно каждый переход в S. Он имеет один из видов: дискретный переход (не содержащий изменений переменных I), непрерывный переход, дискретный переход с произвольным изменением переменных набора I и без изменения локации.

1. Дискретный переход, не содержащий изменений переменных набора I.

Из выполнения свойств ★ для начального состояния рассматриваемого перехода в S и конечного состояния последнего добавленного перехода в S', а также из отсутствия в действиях перехода изменений переменных набора I следует, что в результате применения действий рассматриваемого перехода (изменение внутренних переменных и обнуление таймеров) к предшествующему этому переходу состоянию в S и к конечному состоянию последнего добавленного перехода в S' будут получены состояния, для которых также выполнены свойства ★. Из выполнения для полученных состояний свойства 3 ★ и из истинности инварианта локации-назначения рассматриваемого перехода в S следует истинность инварианта локации-назначения (после выполнения действий перехода) в S'.

Добавим в S' переход, соответствующий рассматриваемому переходу в S, и перейдем к следующему переходу в S.

2. Непрерывный переход. Так как значения таймеров не могут сравниваться со значениями переменных I (следовательно, возможность продвижения времени на некоторую величину не

зависит от значений этих переменных), и для текущего состояния вычисления S' и начального состояния рассматриваемого перехода в S выполнено свойство 2 ★, то из истинности инварианта текущей локации в конечном состоянии рассмотренного перехода в S следует истинность этого инварианта после выполнения соответствующего перехода в S'. Поэтому непрерывный переход, соответствующий рассматриваемому переходу из текущего состояния вычисления S, возможен из текущего состояния S'. Добавим такой переход в S' и перейдем к следующему переходу. При этом для полученного состояния S' и конечного состояния рассматриваемого перехода в S будут выполняться свойства ★, поскольку они выполнялись для состояния, предшествующего рассматриваемому переходу в S, и для состояния, предшествующего добавленному переходу в S'. Поэтому любой переход, возможный из конечного состояния рассмотренного перехода в S, возможен и из конечного состояния добавленного перехода в S'.

3. Дискретный переход с некоторым изменением переменных набора I и без изменения локации.

Если в полученном в результате этого изменения наборе значений переменных I не содержится чисел, превышающих k, добавим такой переход в S' и перейдем к следующему переходу. Для конечного состояния этого перехода в S' и конечного состояния рассмотренного перехода в S выполняются все свойства ★: выполнение свойства 1 следует из вида перехода; свойств 2 и 3 — из совпадения значений переменных I в конечном состоянии рассмотренного перехода в S и в конечном состоянии добавленного перехода в S' и из выполнения этих свойств для состояния, предшествующего рассматриваемому переходу в S, и для состояния, предшествующего добавленному переходу в S'. Следовательно, для конечного состояния этого перехода в S' значение инварианта соответствующей локации совпадает со значением инварианта этой локации в конечном состоянии рассматриваемого перехода в S и равно $True$, и любой переход, возможный из конечного состояния рассматриваемого перехода в S, возможен и из конечного состояния добавленного перехода в S'.

Пусть, напротив, полученный в результате этого изменения набор N_1 значений переменных I содержит некоторое количество чисел, превышающих k. Тогда сформируем набор N_2 значений переменных I в точности так же, как формируется набор начальных значений переменных I для вычисления S', если набор начальных значений переменных I в вычислении S содержит числа, большие k. Добавим в S' переход, в результате которого переменные I принимают значения N_2, а локация не изменяется. Для конечного состояния этого перехода в S' и конечного состояния рассмотренного перехода в S выполняются все свойства ★ (обоснование этого совпадает с обоснованием аналогичного факта для начальных состояний S и S'). Поэтому для конечного состояния этого перехода в S' значение инварианта соответствующей локации совпадает со значением инварианта этой локации в конечном состоянии рассматриваемого перехода в S и равно $True$, и любой переход, возможный из конечного состояния рассматриваемого перехода в S, возможен и из конечного состояния добавленного перехода в S'.

Таким образом будем добавлять из S переходы в S' до тех пор, пока в S не будет достигнута локация E. Локация E будет достигнута и в S', так как в S имеется дискретный переход в эту локацию, а каждому переходу в S соответствует переход с той же локацией назначения в S'. При этом в вычислении S' последовательности значений переменных набора I не содержат чисел, больших k.

Это противоречит предположению, следовательно, утверждение 9 доказано.

Если исходный параметризованный автомат и автомат-наблюдатель имеют k общих целочисленных переменных, удовлетворяющих условиям 1 и 3 утверждения 9, то для этих k переменных и автомата, эквивалентного сети из этих двух автоматов, выполнены все условия утверждения 9:

– условия 1 и 3 выполнены в связи с тем, что при преобразовании сети автоматов к одному эквивалентному автомату не порождаются выражения с новыми операторами присваивания и сравнения;
– выполнение условия 2 следует из свойств автомата-наблюдателя (обоснование аналогично обоснованию выполнения условия 7 на стр. 168).

В этом случае при проверке выполнения для исходного автомата требования, которому соответствует автомат-наблюдатель, достаточно ограничиться значениями рассматриваемых переменных, не превышающими k.

Б.6. Совместное применение доказанных утверждений

Рассмотрим, как доказанные утверждения могут быть применены совместно к автоматам общего вида в целях сокращения диапазонов значений параметров и переменных, перебираемых при верификации. Согласно разделу Б.1 под автоматами общего вида понимаются параметризованные автоматы, которые могут иметь временные параметры, параметры, задающие размеры массивов, индексные параметры и переменные (как внутренние, так и переменные интерфейса). Других видов параметров в рассматриваемых автоматах общего вида нет.

При доказательстве утверждений 3—5 для простоты рассматривались автоматы, не имеющие переменных (это ограничение было введено на стр. 142). Сформулируем и докажем аналогичные утверждения, справедливые для автоматов общего вида, применимые совместно с утверждениями 6—10.

Ниже используются следующие обозначения:

– $cond\_v(\overline{v})$ — логическое выражение, значение которого определяется однозначно значениями переменных параметризованного автомата;
– $cond\_t(\overline{t})$ — логическое выражение, значение которого определяется однозначно значениями таймеров и параметров параметризованного автомата.

При этом в выражениях вида $cond\_v(\overline{v})$, помимо переменных, могут использоваться константы и функции, значения которых определяются однозначно значениями переменных параметризованного автомата. В выражениях вида $cond\_t(\overline{t})$, помимо таймеров и параметров, могут использоваться константы и функции, значения которых определяются однозначно значениями таймеров и параметров параметризованного автомата.

Лемма 5. Пусть параметризованный автомат A общего вида либо не имеет переменных, либо имеет переменные и удовлетворяет следующим условиям:

1. Все условия переходов и инварианты локаций, включающие одновременно таймеры и переменные, имеют вид $cond\_v(\overline{v})\&\&cond\_t(\overline{t})$.
2. Изменение переменных выполняется на переходах, имеющих один из видов:
   а) Дискретный переход с такими действиями, что для любого набора значений переменных $\overline{v_0}$ однозначно определен набор значений переменных $\overline{v_1}$, получающийся в результате выполнения этих действий над $\overline{v_0}$ (пример действий, для которых такое условие однозначности НЕ выполняется: $t == p?v = 0 : v = 1$; — конечное значение переменной v зависит от значения таймера t).
   б) Безусловный дискретный переход из некоторой локации в нее же с произвольным изменением некоторых переменных.

Пусть также задана локация E этого автомата. Тогда для параметризованного автомата A найдется такой параметризованный автомат B без переменных, а для локации E автомата A найдется такой набор локаций автомата B, что достижимость локации E автомата A эквивалентна достижимости хотя бы одной локации из этого набора локаций параметризованного автомата B.

Замечание. На странице 142 аналогичный вывод сделан для экземпляров автоматов. Для экземпляров автоматов он очевиден и непосредственно следует из правил построения для экземпляра автомата с переменными эквивалентного экземпляра автомата без переменных.

Доказательство.

Для параметризованных автоматов без переменных утверждение леммы 5 очевидно.

Пусть параметризованный автомат A имеет переменные и удовлетворяет условиям 1, 2 леммы 5. Построим искомый параметризованный автомат B без переменных.

По определению временных автоматов с остановкой таймеров (см. раздел 2.2), значения каждой переменной принадлежат конечному множеству целых чисел.

Множество локаций автомата B определяется согласно правилам со стр. 142, т.е. каждой локации L исходного автомата соответствует набор локаций $\langle L, \overline{v_j}\rangle$, $j \in \overline{1, N_V}$, где $\overline{v_j}$ — j-й набор значений переменных, N_V — количество всевозможных наборов значений переменных. Начальная локация автомата B — локация $\langle L_0, \overline{v_0}\rangle$, где L_0 — начальная локация автомата A, $\overline{v_0}$ — начальные значения переменных.

Наборы таймеров и параметров автомата B совпадают с наборами таймеров и параметров автомата A. Если некоторая локация L автомата A является срочной (приоритетной), то все соответствующие локации автомата B являются срочными (приоритетными).

В автомате A для некоторого таймера t условие активности в некоторой локации L может иметь один из двух видов:

1. $cond\_v(\overline{v})$. В этом случае в автомате B условие активности этого таймера тождественно истинно в каждой локации $\langle L, \overline{v_j}\rangle$, где $cond\_v(\overline{v_j}) = True$, и тождественно ложно в каждой локации $\langle L, \overline{v_j}\rangle$, где $cond\_v(\overline{v_j}) = False$;
2. Логическая константа $True$ или $False$. В этом случае в автомате B условие активности этого таймера в каждой локации $\langle L, \overline{v_j}\rangle$ равно той же логической константе.

В автомате A инвариант некоторой локации L может иметь один из следующих видов:

1. $cond\_v(\overline{v})\&\&cond\_t(\bar{t})$. В этом случае в автомате B каждая локация $\langle L, \overline{v_j}\rangle$ имеет инвариант, равный булевой константе $False$, если $cond\_v(\overline{v_j}) = False$, и равный $cond\_t(\bar{t})$ иначе.
2. $cond\_v(\overline{v})$. В этом случае в автомате B каждая локация $\langle L, \overline{v_j}\rangle$ имеет инвариант, равный булевой константе $True$, если $cond\_v(\overline{v_j}) = True$, и равный булевой константе $False$, если $cond\_v(\overline{v_j}) = False$.
3. $cond\_t(\bar{t})$. В этом случае в автомате B каждая локация $\langle L, \overline{v_j}\rangle$ имеет инвариант $cond\_t(\bar{t})$.
4. Логическая константа $True$ или $False$. В этом случае в автомате B каждая локация $\langle L, \overline{v_j}\rangle$ имеет инвариант, равный той же булевой константе.

Каждому переходу автомата A из локации L_1 в локацию L_2 с условием вида $cond\_v(\overline{v})\&\&cond\_t(\bar{t})$ соответствует набор переходов в автомате B из локаций $\langle L_1, \overline{v_j}\rangle$, таких что $cond\_v(\overline{v_j}) = True$, в локации $\langle L_2, \overline{v_i}\rangle$, такие что $\overline{v_i}$ получается из $\overline{v_j}$ при выполнении действий рассматриваемого перехода. Поскольку рассматриваемый переход в автомате A не является безусловным, то он имеет вид 2.а (см. условие леммы 5). Следовательно, набор $\overline{v_i}$ определяется однозначно набором $\overline{v_j}$ и действиями рассматриваемого перехода, поэтому в автомате B локация назначения соответствующего перехода однозначно определяется локацией-источником и действиями рассматриваемого перехода. Все переходы в автомате B, соответствующие рассматриваемому переходу, имеют условие $cond\_t(\bar{t})$. Аналогичным образом в автомате B формируются переходы, соответствующие переходам вида 2.а автомата A с условиями вида $cond\_v(\overline{v})$ (в этом случае все соответствующие переходы в B являются безусловными), вида $cond\_t(\bar{t})$ и вида $True$. В последних двух случаях одному переходу в A будет соответствовать N_V переходов в B — т.е. переходы для всевозможных наборов значений переменных в локации-источнике перехода; значения переменных в локации-назначении перехода однозначно определяются значениями в локации-источнике и действиями на переходе. Каждому переходу вида 2.б из локации L в локацию L автомата A соответствует N_V^2 безусловных переходов в автомате B, т.е. переходы для всевозможных наборов значений переменных в локации-источнике перехода и всевозможных наборов значений переменных в локации-назначения перехода.

Фиксированный набор значений временных параметров задает экземпляр автомата A и соответствующий ему экземпляр автомата B.

Для любого экземпляра X_A автомата A и соответствующего ему экземпляра X_B автомата B верно, что в X_A имеется переход из некоторого состояния $\langle l_A, \overline{v_A}, \overline{c_A}\rangle$ в состояние $\langle {l'}_A, \overline{v_A}', \overline{c_A}'\rangle$, тогда и только тогда, когда в X_B имеется переход из состояния $\langle\langle l_A, \overline{v_A}\rangle, \overline{c_A}\rangle$ в состояние $\langle\langle {l'}_A, \overline{v_A}'\rangle, \overline{c_A}'\rangle$. Для непрерывных переходов это следует из вида инвариантов локаций, для дискретных — из вида инвариантов локаций и переходов между локациями. Поэтому любому вычислению X_A можно взаимно однозначно поставить в соответствие некоторое вычисление X_B. Значит, локация E автомата A достижима тогда и только тогда, когда достижима хотя бы одна из соответствующих ей локаций в автомате B. Лемма 5 доказана.

Напомним, что для автомата, имеющего переменные, недостижимость локации подразумевает ее недостижимость для любых последовательностей значений переменных. Для автоматов, удовлетворяющих лемме 5, такие последовательности формируются при выполнении переходов вида 2.а и 2.б условия леммы 5.

Из леммы 5 и утверждений 3, 4, 5 следуют соответственно утверждения 10, 11, 12, снимающие ограничение отсутствия переменных в автомате.

Утверждение 10. Пусть параметризованный автомат общего вида имеет k временных параметров и удовлетворяет всем условиям утверждения 3 и леммы 5. Тогда для любой локации этого автомата верно, что если она не достижима при любых целых неотрицательных значениях временных параметров, не превышающих k, то она не достижима при любых целых неотрицательных значениях временных параметров.

Доказательство.

Поскольку заданный параметризованный автомат удовлетворяет условиям леммы 5, то для него можно построить эквивалентный с точки зрения достижимости локаций параметризованный автомат, не имеющий переменных. Согласно правилам построения этого эквивалентного автомата, приведенным в доказательстве леммы 5, все используемые в этом автомате выражения, содержащие временные параметры и таймер, имеются и в исходном параметризованном автомате. Т.к. утверждение 3 накладывает ограничения только на вид таких выражений, то из выполнения условий утверждения 3 для исходного параметризованного автомата следует и их выполнение для эквивалентного параметризованного автомата без переменных. Следовательно, для проверки достижимости некоторой локации в этом параметризованном автомате без переменных достаточно проверить ее достижимость для значений временных параметров, не превышающих k. Поэтому и для проверки достижимости некоторой локации в исходном параметризованном автомате достаточно проверить ее достижимость для значений временных параметров, не превышающих k. Утверждение 10 доказано.

Модифицируем ограничения определения класса Л1 со стр. 149 в части ограничений на вид условий переходов следующим образом: пусть либо условие перехода равно логической константе $True$, либо имеет вид $cond(\overline{v})$, либо представляет собой конъюнкцию выражения вида $cond(\overline{v})$ и элементарных сравнений таймеров с операциями $\{<, <=, ==\}$, содержащую хотя бы одно сравнение вида $t_h == p_j$, где t_h активен в локации-источнике перехода, p_j — один из временных параметров автомата. Ограничения на вид инвариантов оставим прежними. Также введем ограничения на вид переходов, содержащих действия по изменению переменных, совпадающие с ограничениями п. 2 леммы 5. Полученные в результате модификации ограничения обозначим (**). Эти ограничения являются более жесткими, чем ограничения леммы 5, и из их выполнения для некоторого параметризованного автомата следует выполнение для этого параметризованного автомата ограничений леммы 5.

Утверждение 11. Пусть параметризованный автомат общего вида удовлетворяет всем условиям утверждения 4, за исключением условия 5 (требующего выполнения ограничений определения класса Л1), вместо которого выполняются ограничения (**). Тогда для любой локации этого автомата верно, что если она не достижима при любых целых неотрицательных значениях временных параметров, не превышающих 22, то она не достижима при любых целых неотрицательных значениях временных параметров.

Доказательство.

Поскольку для рассматриваемого параметризованного автомата выполняются ограничения (**), то для него выполняются ограничения леммы 5, и для него можно построить эквивалентный

с точки зрения достижимости локаций параметризованный автомат, не имеющий переменных. Согласно правилам построения переходов в этом эквивалентном автомате и ограничениям (**), каждое условие перехода в этом автомате либо равно константе $True$, либо представляет собой конъюнкцию элементарных сравнений таймеров с операциями $\{<, <=, ==\}$, содержащую хотя бы одно сравнение вида $t_h == m_j$, где t_h активен в локации-источнике перехода, m_j — значение одного из временных параметров автомата, то есть для построенного автомата выполняются ограничения определения класса Л1 со стр. 149. Таким образом, для построенного параметризованного автомата, не имеющего переменных, выполняются условия утверждения 4. Поэтому для любой его локации верно, что если она не достижима при любых целых неотрицательных значениях временных параметров, не превышающих 22, то она не достижима при любых целых неотрицательных значениях временных параметров. Следовательно, это верно и для любой локации исходного автомата, имеющего переменные. Утверждение 11 доказано.

Утверждение 12. Пусть параметризованный автомат общего вида имеет хотя бы один неостанавливаемый таймер и ровно один временной параметр p. Пусть в сравнениях с таймерами может использоваться только параметр p, и, возможно, константа 0, а также выполнены условия леммы 5. Тогда для любой локации автомата верно, что если она не достижима при $p = 0$ и $p = 1$, то она не достижима при любых целых неотрицательных значениях параметра p.

Доказательство утверждения 12 аналогично доказательству утверждения 10 с тем лишь отличием, что параметризованный автомат без переменных, эквивалентный исходному автомату, удовлетворяет условиям утверждения 5, а не утверждения 3.

Утверждения 8—12 определяют подходы к выбору достаточных для рассмотрения при верификации автоматов диапазонов значений временных и индексных параметров, а также переменных интерфейса. При этом формулировки и доказательства этих утверждений не накладывают ограничения на «принадлежность» этих параметров и переменных: они могут как быть «самостоятельными», так и являться элементами массивов. В утверждениях 6 и 7 описана схема выбора достаточных для верификации диапазонов значений параметров-размеров массивов. Если для некоторого параметризованного автомата выполнены условия одного из утверждений 6, 7, то для проверки выполнения того или иного требования к этому автомату при всевозможных значениях размера используемых в автомате массивов достаточно проверифицировать этот автомат при всевозможных значениях размера массивов, не превышающих некоторого k (значение k определяется согласно соответствующему утверждению). Если элементами массивов являются временные или индексные параметры, или переменные интерфейса, то достаточные для верификации диапазоны их значений определяются согласно утверждениям 8—12 с учетом диапазонов размеров массивов, полученных согласно утверждениям 6, 7. Пусть, например, параметризованный автомат работает с массивом переменных интерфейса (других переменных интерфейса нет) и удовлетворяет условиям утверждений 6 и 9. Согласно утверждению 6, верификацию этого автомата достаточно выполнить для массивов размеров, не превышающих некоторого k. То есть достаточно выполнить верификацию параметризованных автоматов, работающих с массивом переменных интерфейса размера 1, размера 2, ..., размера k. Тогда, согласно утверждению 9, при верификации параметризованного автомата, работающего с массивом переменных интерфейса размера 1, достаточно рассматривать значения единственной переменной интерфейса, не превышающие 1; с массивом

переменных интерфейса размера 2 — значения этих переменных, не превышающие 2; и т.д., с массивом переменных размера k — значения этих переменных, не превышающие k. Итого, верификацию этого автомата достаточно выполнить для массивов размеров, не превышающих k, и для значений переменных интерфейса, не превышающих k.

Таким образом, получен набор утверждений 6—12, в которых рассматриваются автоматы общего в рамках данной работы вида, имеющие параметры всех используемых в работе видов (временные, индексные, задающие размеры массивов) и переменные.

Если условия одного утверждения не противоречат условиям другого утверждения, то эти утверждения могут быть применены к одному и тому же параметризованному автомату. В противном случае существовали бы автоматы, для которых выполнены все условия некоторого утверждения, а само утверждение не выполнено.

Докажем, например, что могут быть совместно применены утверждения 9 (о выборе диапазонов значений переменных) и 10 (о выборе диапазонов значений временных параметров). Предположим, что это неверно, т.е. существует параметризованный автомат, удовлетворяющий условиям этих утверждений, имеющий k временных параметров и m переменных интерфейса, такой что некоторая его локация E недостижима при любых значениях временных параметров, не превышающих k, и при любых наборах последовательностей значений переменных интерфейса, таких что каждое значение не превышает m, но достижима при других значениях временных параметров и/или при других наборах последовательностей значений переменных интерфейса. Рассмотрим экземпляр A этого автомата и набор последовательностей значений переменных, при которых достижима локация E. Возможно два случая:

1. В наборе последовательностей значений переменных интерфейса хотя бы одно значение хотя бы одной из переменных превышает m. Заменим в рассматриваемом параметризованном автомате временные параметры на константы, являющиеся значениями этих параметров в экземпляре автомата A. Получим параметризованный автомат (возможно, не имеющий параметров, если других параметров у автомата нет), удовлетворяющий условиям утверждения 9, для которого само утверждение 9 нарушено (т.е. получено противоречие).
2. В наборе последовательностей значений переменных интерфейса ни одно из значений переменных не превышает m, а значение одного из временных параметров превышает k. Рассмотрим параметризованный автомат, отличающийся от исходного автомата лишь тем, что диапазон значений каждой его переменной интерфейса равен $[0,m]$. Для этого автомата выполнены все условия утверждения 10, а само утверждение 10 нарушено: его локация E недостижима при любых значениях временных параметров, не превышающих k, и при *любых* наборах последовательностей значений переменных интерфейса (см. замечание после доказательства леммы 5 на стр. 203), но достижима для некоторого набора значений временных параметров, в котором хотя бы одно значение превышает k (т.е. получено противоречие).

Таким образом, введенное предположение неверно, и утверждения 9 и 10 могут быть применены совместно. Аналогичным образом доказывается совместимость других утверждений, условия которых не противоречат друг другу.

В таблице Б.1 содержится информация о совместимости условий утверждений. Утверждения 1—5 не приведены в таблице, т.к. они являются вспомогательными и непосредственно к автоматам-моделям и автоматам-наблюдателям не применяются.

Таблица Б.1 — Совместимость условий утверждений

Номер утверждения	6	7	8	9	10	11	12
6	x	−	+	+	+	+	+
7	−	x	+	−	+	−	+
8	+	+	x	+	+	+	+
9	+	−	+	x	+	+	+
10	+	+	+	+	x	−	−
11	+	−	+	+	−	x	−
12	+	+	+	+	−	−	x

Если для верификации некоторого автомата необходимо применить утверждения, условия которых противоречат друг другу, то рекомендуется выполнить декомпозицию этого автомата на несколько более простых автоматов.

Итого, при проверке выполнения некоторого требования для автомата-модели компонента МВС необходимо выполнить действия в соответствии со схемой, приведенной на рисунке Б.6. Записи вида «Утв. X?» на рисунке означают проверку выполнения условий утверждения X для автомата-модели, вида «Объекты X?» — проверку наличия в автомате-модели объектов типа X (временные параметры, переменные и т.д.). Запись «Периодические автоматы?» означает проверку периодичности автомата-модели и автомата-наблюдателя (для автоматов-наблюдателей, построенных в данной работе, периодичность определяется в соответствии с таблицей Б.2). Для всех построенных в данной работе автоматов-наблюдателей условия соответствующих утверждений выполнены, номера выполненных утверждений приведены в таблице Б.2.

Таблица Б.2 — Соответствие разработанных автоматов-наблюдателей и утверждений, условия которых для них выполнены

Базовый тип автоматов	Утверждения, условия которых выполнены для автоматов-наблюдателей	Периодичность автоматов-наблюдателей
T (модель функциональной задачи)	Утверждения 6, 11	Да
TS (модель планировщика задач раздела)	Утверждения 6, 9	Нет
CS (модель планировщика ядра)	Следствие утверждения 7; утверждения 8, 10	Да
L (модель виртуального канала)	Утверждение 12	Нет

Для утверждений 6 и 9 схема получения диапазонов значений параметров/переменных на основе характеристик автомата-модели и автомата-наблюдателя приведена после доказательства этого утверждения на стр. 177 и 201 соответственно. Для утверждения 7, следствия из утверждения 7 и утверждения 8 схема получения таких диапазонов приведена непосредственно в этих утверждениях и следствии. Для утверждений 10—12 схема получения таких диапазонов совпадает со схемой для утверждений 3—5, которая приведена в разделе Б.2 (см. стр. 160 и далее).

Условия приведенных в данном приложении утверждений относятся к структуре и синтаксису автоматов. Проверка выполнения таких условий может быть автоматизирована. Если для некоторого автомата условия не выполняются, то необходимо либо модифицировать автомат так, чтобы условия выполнялись, либо проводить верификацию для всех возможных значений параметров и переменных при явном введении ограничений на диапазоны этих значений, например, исходя из ограничений на моделируемую МВС. Для всех разработанных автором автоматов-моделей и автоматов-наблюдателей условия соответствующих утверждений выполняются.

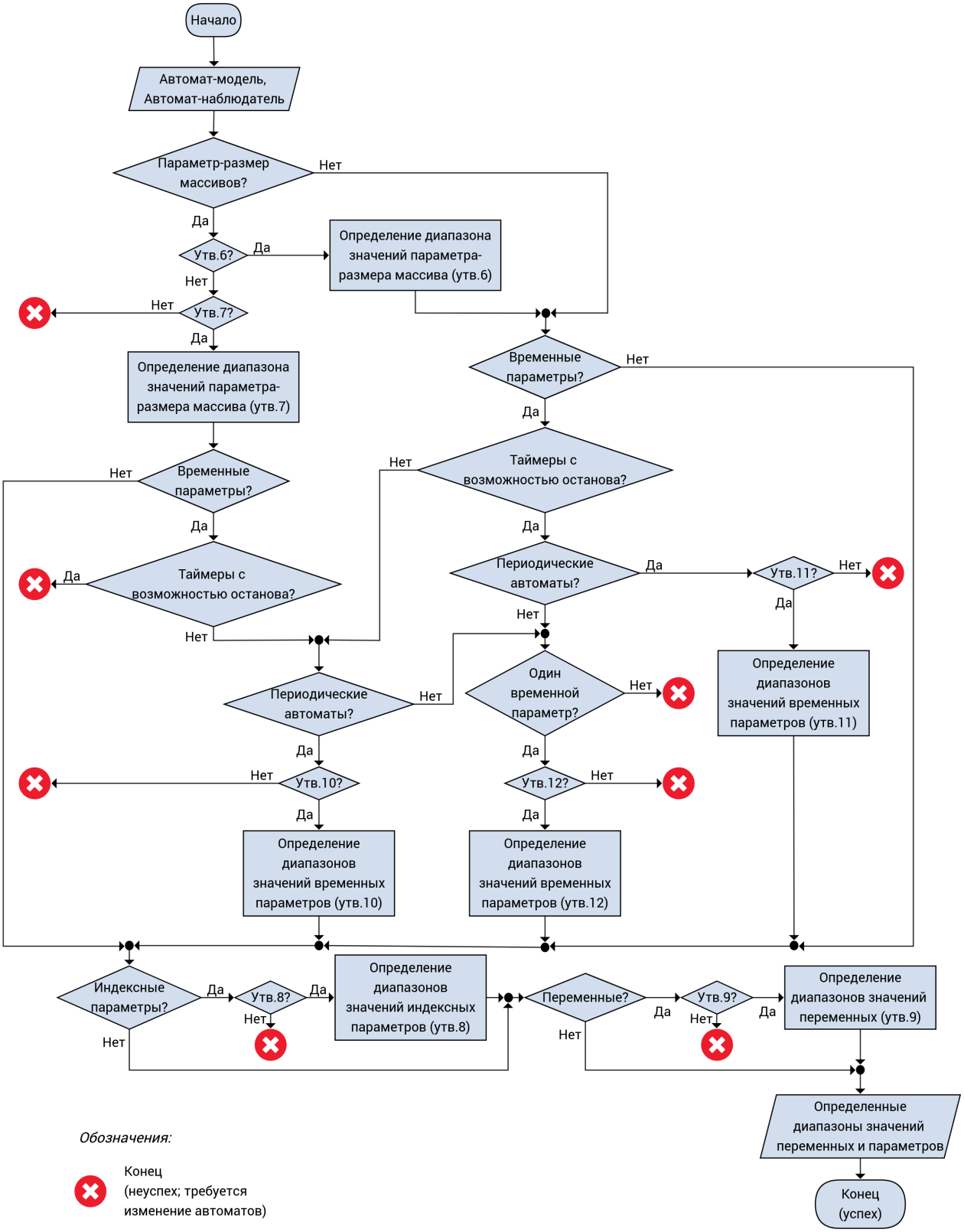

Рисунок Б.6 — Последовательность действий по определению достаточных для верификации диапазонов значений параметров и переменных

Приложение В

Автоматы-наблюдатели для проверки выполнения требований корректности к моделям компонентов МВС

В разделе 3.2.2 приведены формулировки требований к моделям компонентов МВС, в том же разделе для требования 1 к моделям планировщиков раздела описаны принципы построения автомата-наблюдателя и приведен сам автомат-наблюдатель. В настоящем приложении приведены автоматы-наблюдатели для проверки выполнения остальных требований к моделям компонентов МВС.

В.1. Автоматы для требований к моделям планировщиков раздела

Формулировка требования 1 и автомат-наблюдатель, соответствующий этому требованию, приведены в разделе 3.2.2.

Для всех автоматов-наблюдателей, описанных в настоящем разделе (В.1), выполняются условия, требующиеся для применения к этим автоматам-наблюдателям (совместно с соответствующими автоматами-моделями) утверждений 6 и 9 приложения Б. Поэтому выбор диапазонов значений параметров и переменных при верификации осуществляется согласно этим утверждениям. Пример применения этих утверждений к конкретной паре из автомата-модели планировщика раздела и автомата-наблюдателя, соответствующего требованию 1, приведен на стр. 71.

Требование 1а. *Для любого раздела верно, что в некоторый момент времени вытеснена может быть только выполняющаяся в этот момент работа.*

Согласно рассуждениям на стр. 69 раздела 3.2.2, автомат-наблюдатель для проверки выполнения требования 1 позволяет проверить в том числе, что при наличии выполняющейся работы вытеснена может быть только эта выполняющаяся работа. Поэтому, если выполнено требование 1, то для проверки выполнения требования 1а достаточно проверить, что *при отсутствии выполняющихся работ никакая работа не может быть вытеснена*.

Автомат-наблюдатель, соответствующий этому требованию, приведен на рисунке В.1 и имеет вид, аналогичный виду автомата-наблюдателя, соответствующего требованию 1. Отличие состоит лишь в том, что переходы между локациями $Busy$, $Idle$ и $ERROR$ соответствуют последовательностям синхронизаций, специфичным для требования 1а.

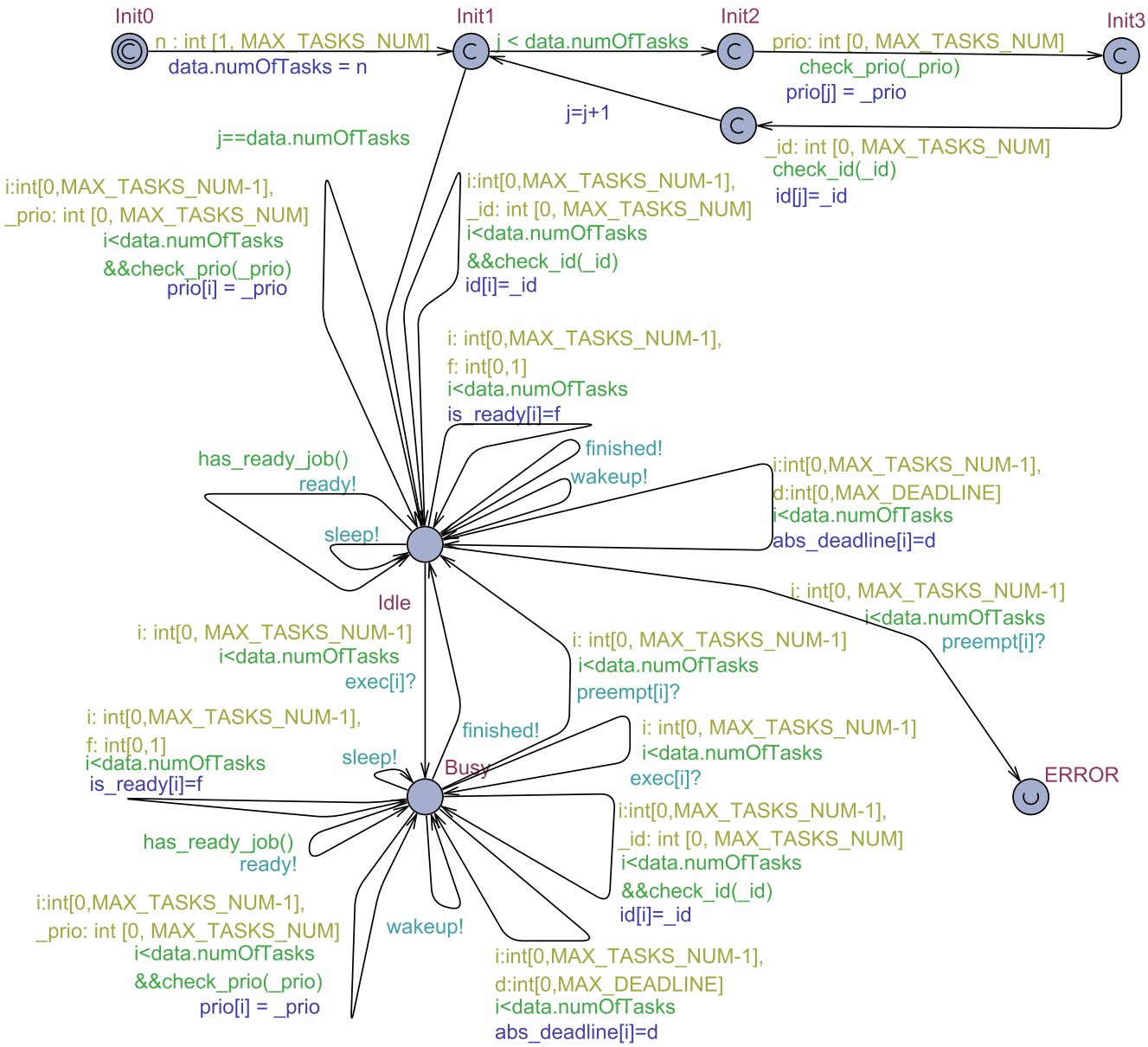

Рисунок В.1 — Автомат-наблюдатель, соответствующий требованию 1а к модели планировщика раздела

Требование 2.1. *Планировщик раздела может поставить работу на выполнение только в рамках окна этого раздела.*

Для модели планировщика раздела окнами этого раздела являются интервалы между синхронизациями по каналам $wakeup$ и $sleep$.

Автомат-наблюдатель, соответствующий требованию 2.1, приведен на рисунке В.2. Помимо локаций $\{Init0, ..., Init3\}$, в автомате-наблюдателе присутствуют локации $Inactive$ и $Active$, соответствующие неактивному (вне окон раздела) и активному (внутри окон) состояниям планировщика, а также «плохая» локация $ERROR$. Аналогично автомату-наблюдателю, соответствующему требованию 1, в автомате-наблюдателе, соответствующем требованию 2.1, для каждой из локаций $Inactive$ и $Active$ имеются переходы в ту же локацию с недетерминированным изменением переменных is_ready, $abs\_deadline$, $prio$, id.

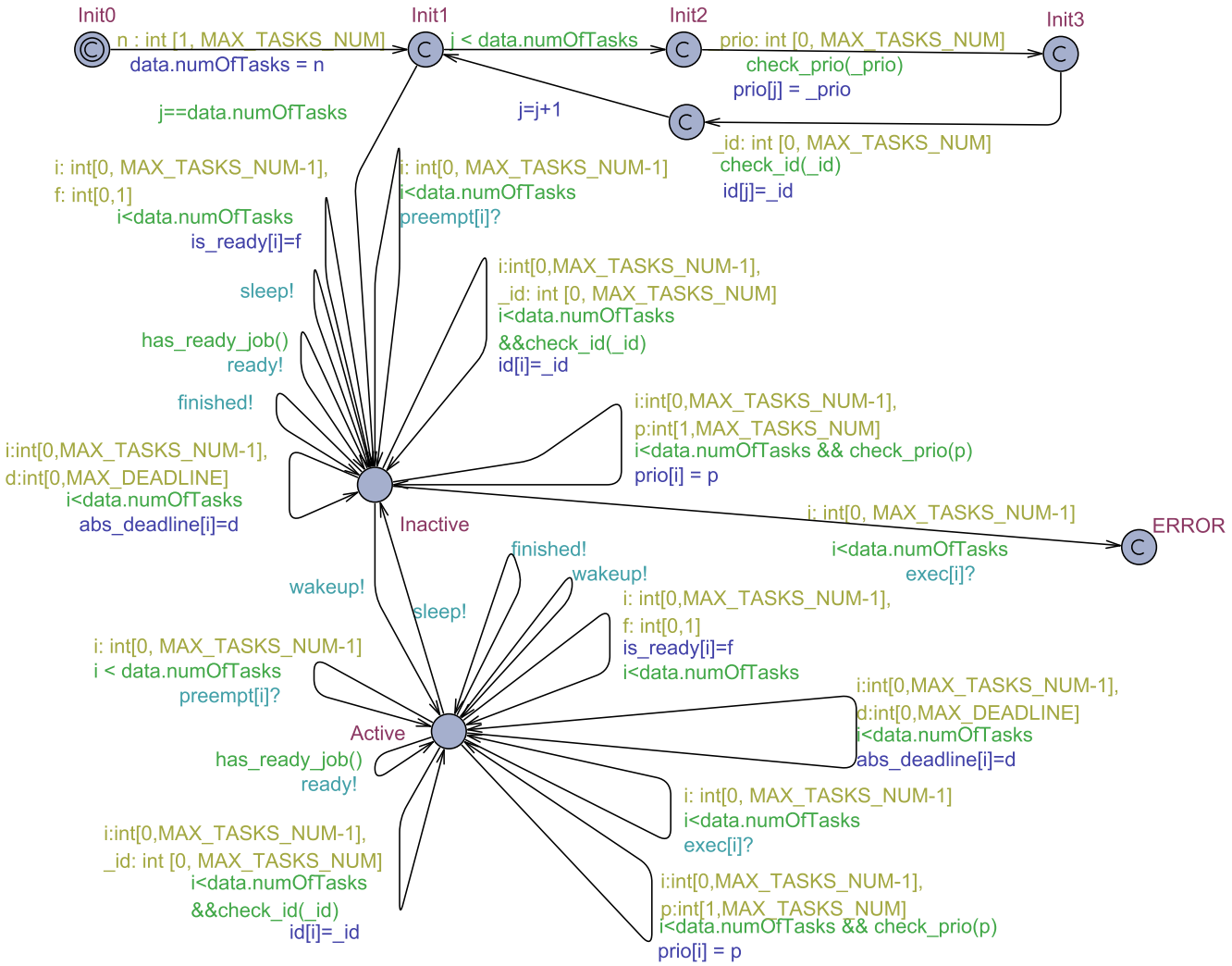

Рисунок B.2 — Автомат-наблюдатель, соответствующий требованию 2.1 к модели планировщика раздела

Требование 2.2. *Для любого раздела верно, что если в некоторый момент времени на вычислителе выполняется работа этого раздела и происходит закрытие текущего окна этого раздела, то некоторая работа должна быть мгновенно вытеснена, либо завершена.*

Автомат-наблюдатель, соответствующий требованию 2.2, приведен на рисунке B.3 и имеет вид, аналогичный виду автомата-наблюдателя, соответствующего требованию 1. Если автомат-наблюдатель находится в локации $Busy$ (выполняется некоторая работа), и происходит синхронизация по каналу $sleep$ (закрытие окна), то флаг $flag$ (обозначающий, что окно закрыто и выполняется работа) становится истинным и обнуляется таймер c. При переходах из локации $Busy$ с синхронизациями по каналам $preempt_i$ и $finished$ флаг $flag$ принимает ложное значение. Если флаг $flag$ имеет истинное значение и значение таймера c больше 0, то это соответствует некорректной последовательности синхронизаций, и, следовательно, выполняется переход в локацию $ERROR$.

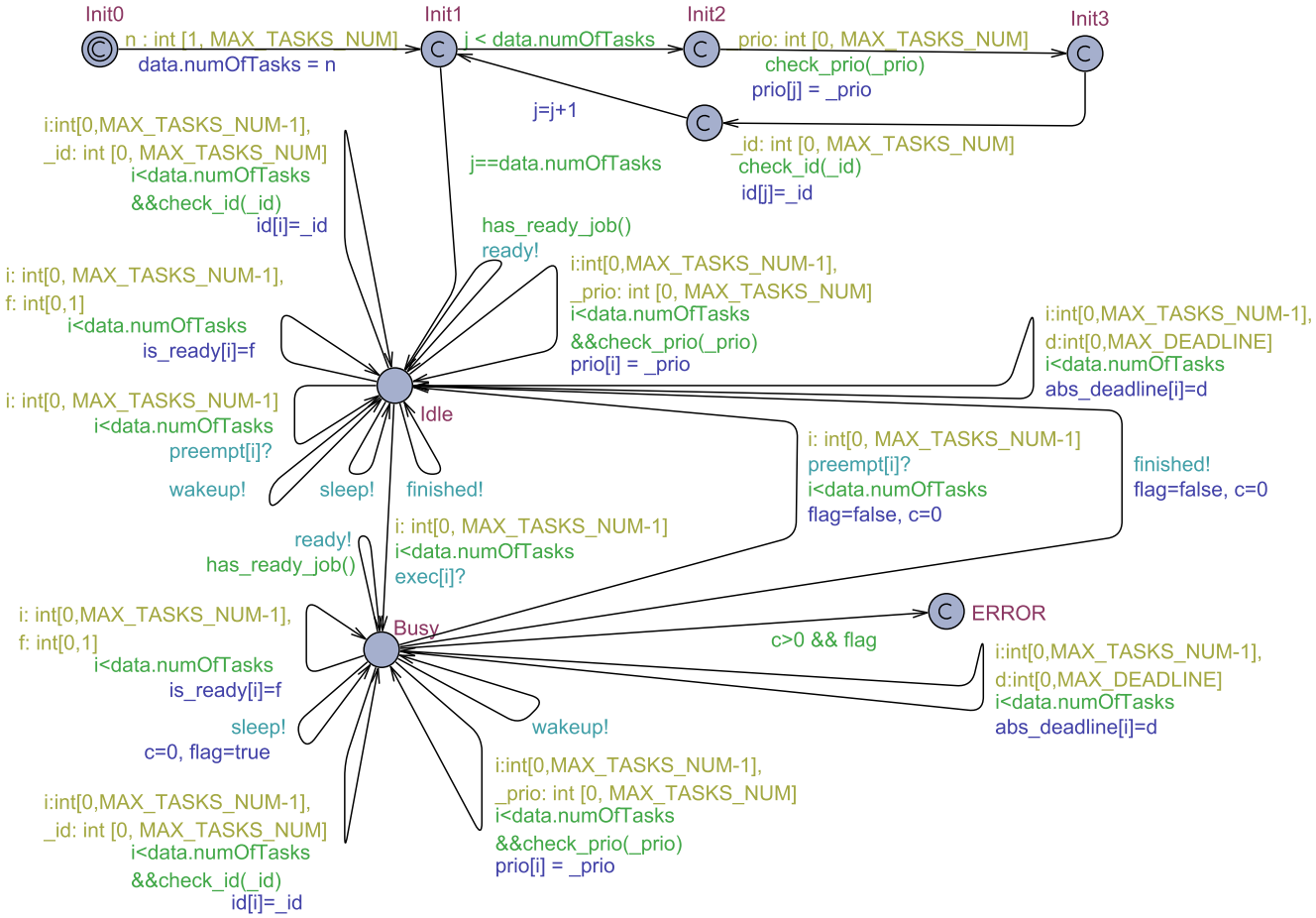

Рисунок В.3 — Автомат-наблюдатель, соответствующий требованию 2.2 к модели планировщика раздела

Требование 3. *Для любого раздела верно, что планировщик данного раздела может поставить на выполнение только готовую работу этого раздела [45].*

Согласно методу построения модели конкретной МВС (раздел 2.5) модель планировщика раздела взаимодействует лишь с моделями тех задач, которые принадлежат этому разделу, и в рамках автомата, моделирующего планировщик раздела, для получения информации о готовности работ используются переменные is_ready_i. Согласно предложенной обобщенной модели МВС (раздел 2.4), модель планировщика раздела может лишь считывать значения этих переменных, но не изменять их.

Автомат-наблюдатель, соответствующий требованию 3, приведен на рисунке В.4. Помимо локаций $\{Init0, ..., Init3\}$, в автомате-наблюдателе присутствует локация $Work$, соответствующая корректному функционированию планировщика, а также «плохая» локация $ERROR$. Аналогично автоматам-наблюдателям, соответствующим предыдущим требованиям, в автомате-наблюдателе, соответствующем требованию 3, из локации $Work$ имеются переходы в ту же локацию с недетерминированным изменением переменных is_ready, $abs\_deadline$, $prio$, id.

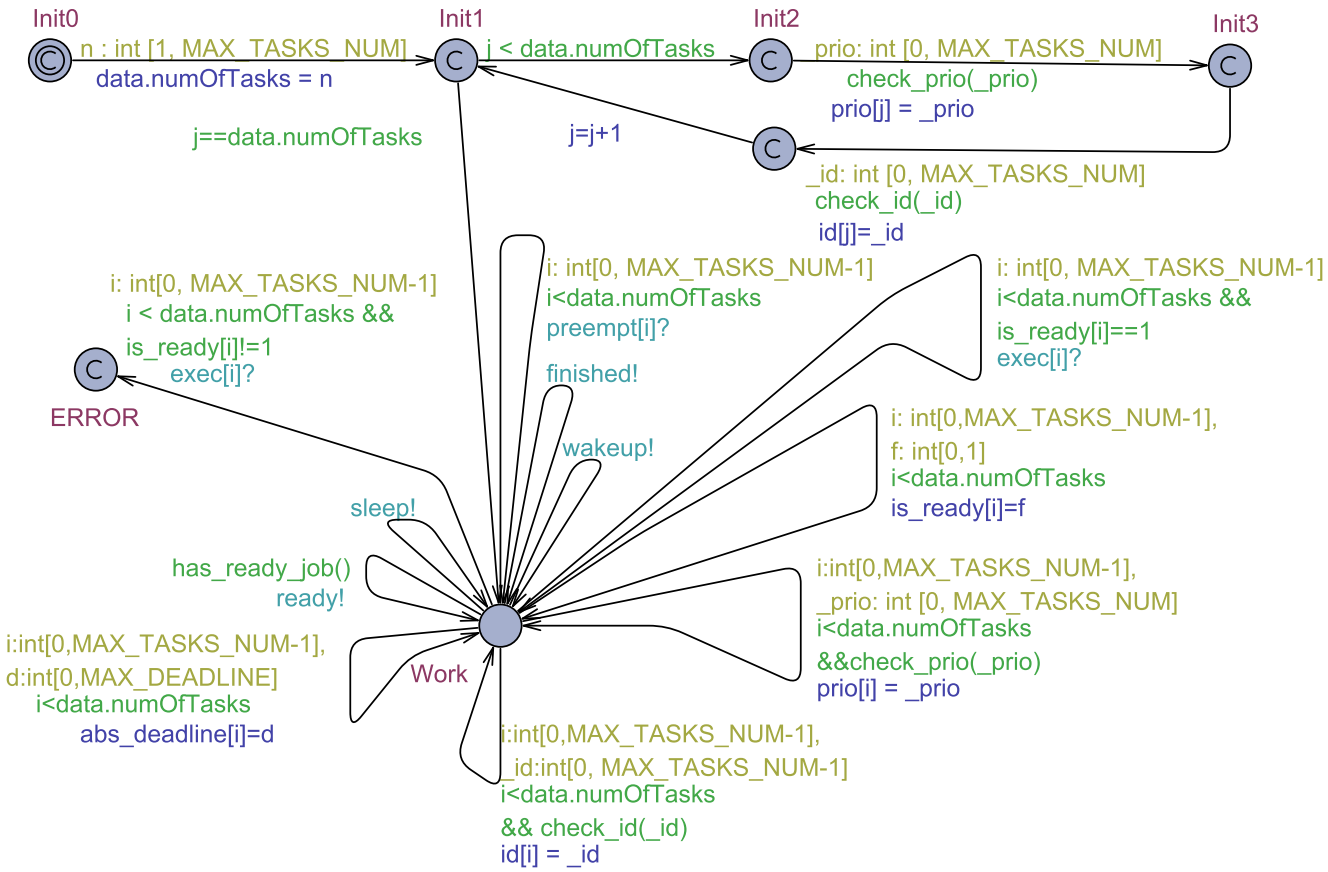

Рисунок В.4 — Автомат-наблюдатель, соответствующий требованию 3 к модели планировщика раздела

Требование 4.1. *Работа может быть поставлена на выполнение только вследствие следующих событий: открытие окна раздела, появление новой готовой работы, завершение некоторой работы. При этом постановка работы на выполнение происходит мгновенно, то есть одновременно с событием-причиной.*

Автомат-наблюдатель, соответствующий требованию 4.1, приведен на рисунке В.5. Помимо локаций $\{Init0, ..., Init3\}$, в автомате-наблюдателе присутствуют локации $CanExec$ и $CanNotExec$, соответствующие состояниям планировщика (допускающему постановку работы на выполнение и не допускающему), а также «плохая» локация $ERROR$. Аналогично автоматам-наблюдателям, соответствующим предыдущим требованиям, в автомате-наблюдателе, соответствующем требованию 4.1, для каждой из локаций $CanExec$ и $CanNotExec$ имеется переход в ту же локацию с недетерминированным изменением переменных is_ready, $abs\_deadline$, $prio$, id. Для измерения задержек между событиями используется таймер c.

Требование 4.2. *Если открывается окно раздела и имеются готовые работы этого раздела, то на выполнение мгновенно должна быть поставлена некоторая работа.*

Автомат-наблюдатель, соответствующий требованию 4.2, приведен на рисунке В.6. Помимо локаций $\{Init0, ..., Init3\}$, в автомате-наблюдателе присутствует локация $Work$, соответствующая функционированию планировщика, локация $MustExec$, соответствующая состоянию планировщика в момент открытия очередного окна, а также «плохая» локация $ERROR$. Для измерения задержки после открытия очередного окна раздела используется таймер c.

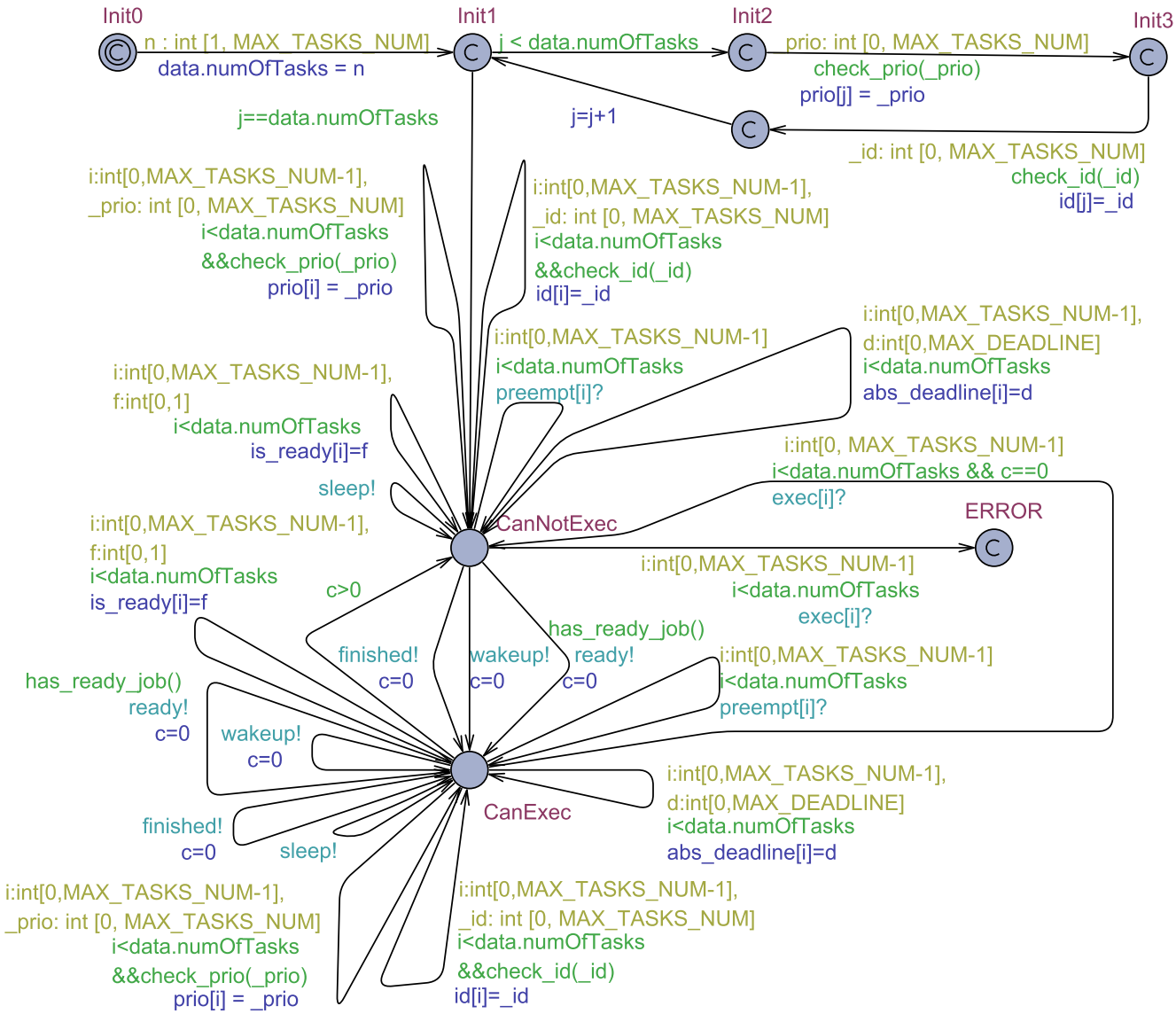

Рисунок В.5 — Автомат-наблюдатель, соответствующий требованию 4.1 к модели планировщика раздела

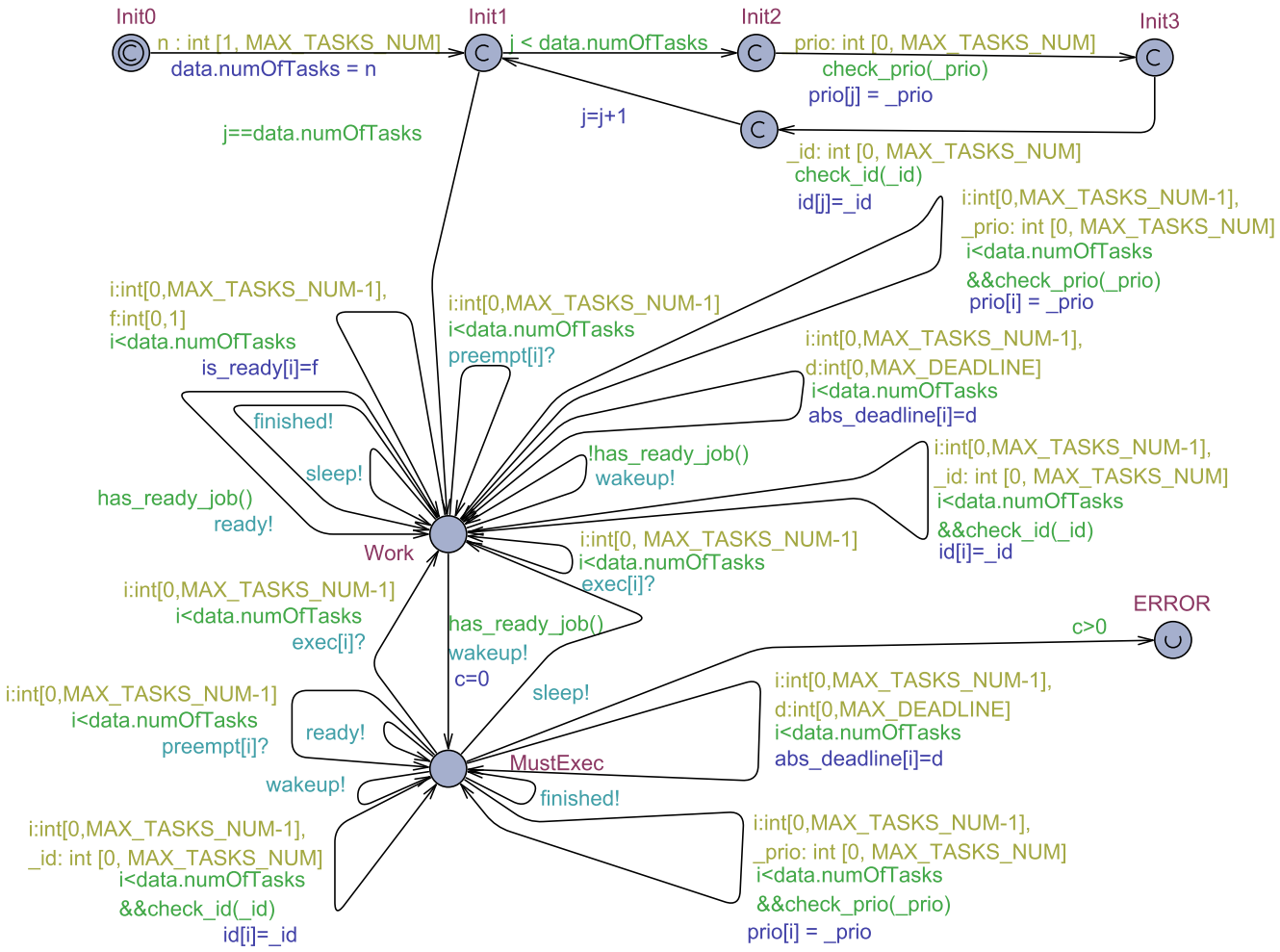

Рисунок В.6 — Автомат-наблюдатель, соответствующий требованию 4.2 к модели планировщика раздела

Требование 4.3. *Если завершилась некоторая работа раздела, имеются готовые работы этого раздела, окно раздела открыто и в данный момент не происходит закрытие окна раздела, то на выполнение мгновенно должна быть поставлена некоторая работа.*

Автомат-наблюдатель, соответствующий требованию 4.3, практически идентичен автомату-наблюдателю, соответствующему требованию 4.2, с тем лишь отличием, что на переходах между локациями синхронизации по каналу $wakeup$ (моделирующие открытие очередного окна раздела) заменены на синхронизации по каналу $finished$ (моделирующие завершение некоторой работы) и наоборот — синхронизации по каналу $finished$ заменены на синхронизации по каналу $wakeup$. Кроме того, в автомате-наблюдателе для проверки требования 4.3 используется переменная is_active, истинное значение которой соответствует тому, что в данный момент окно раздела открыто. Если is_active равна $false$ (окно раздела закрыто), то в случае завершения в этот момент некоторой работы мгновенной постановки на выполнение новой работы не требуется, даже при наличии готовых работ. Автомат-наблюдатель, соответствующий требованию 4.3, приведен на рисунке В.7.

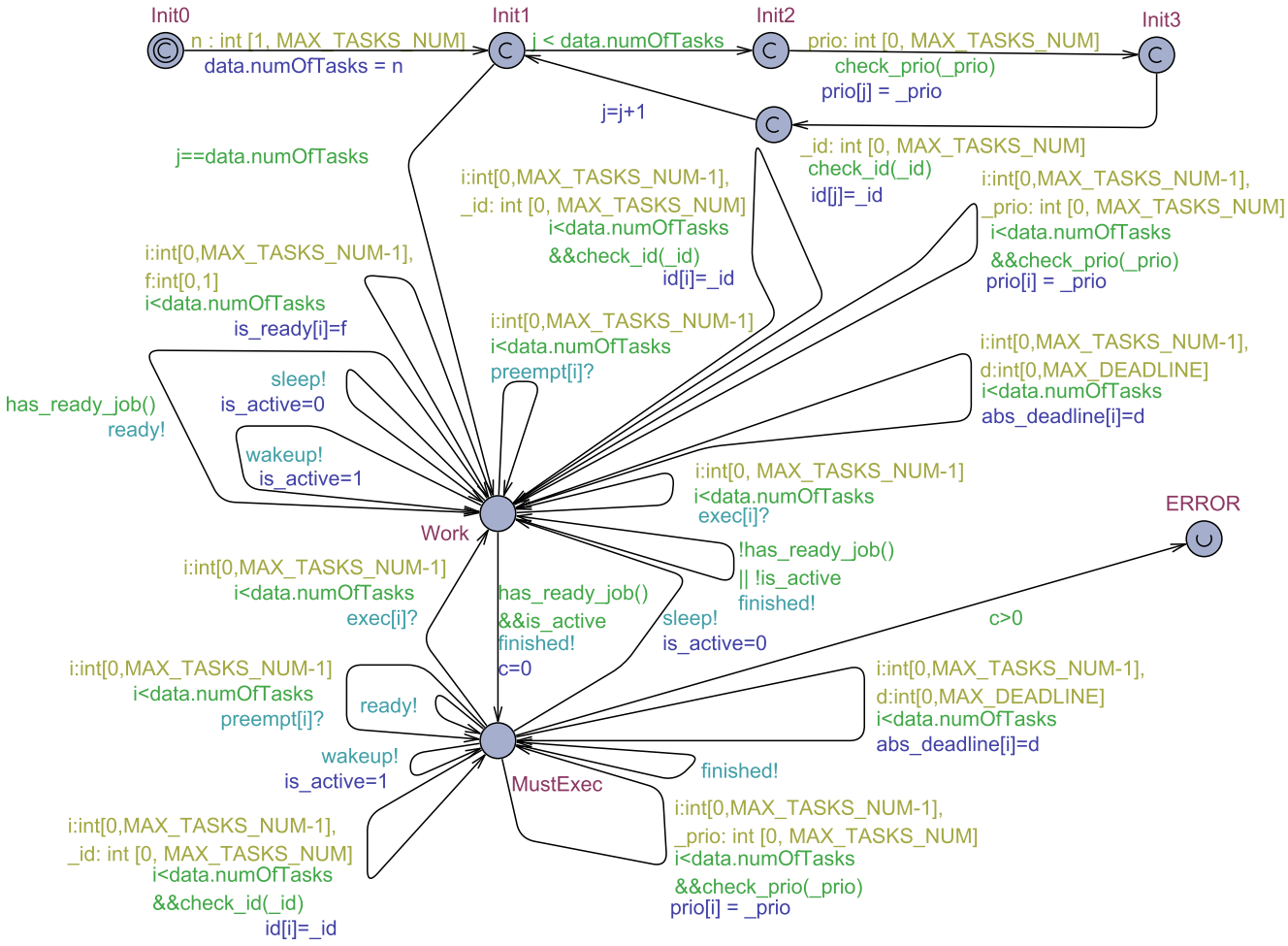

Рисунок В.7 — Автомат-наблюдатель, соответствующий требованию 4.3 к модели планировщика раздела

Требование 5. *Для любого раздела верно, что любая выполняющаяся работа этого раздела может быть вытеснена только в двух случаях: либо при закрытии окна данного раздела, либо при появлении новой готовой работы. При этом вытеснение происходит мгновенно* [45].

Напомним, что при проверке выполнения требования 1 проверяется в том числе и то, что вытеснена может быть только та работа, которая в данный момент выполняется на вычислителе. Для проверки того, что при наличии на вычислителе выполняющейся работы в момент закрытия окна должно происходить вытеснение этой работы, либо завершение, сформулировано требование 7.

Нарушение требования 5 заключается в том, что непосредственно перед вытеснением некоторой работы не происходит ни одно из следующих событий: закрытие окна раздела, появление новой готовой работы.

Автомат-наблюдатель, соответствующий требованию 5, приведен на рисунке В.8. Помимо локаций $\{Init0, ..., Init3\}$, в автомате-наблюдателе присутствуют локации $CanPreempt$ и $CanNotPreempt$, соответствующие состояниям планировщика (допускающему вытеснение работы и не допускающему), а также «плохая» локация $ERROR$. Аналогично автоматам-наблюдателям, соответствующим предыдущим требованиям (например, требованию 4.1), в автомате-наблюдателе, соответствующем требованию 5, для каждой из локаций $CanPreempt$ и $CanNotPreempt$ имеется переход в ту же локацию с недетерминированным изменением переменных is_ready, $abs\_deadline$, $prio$, id. Для измерения задержек между событиями используется таймер c.

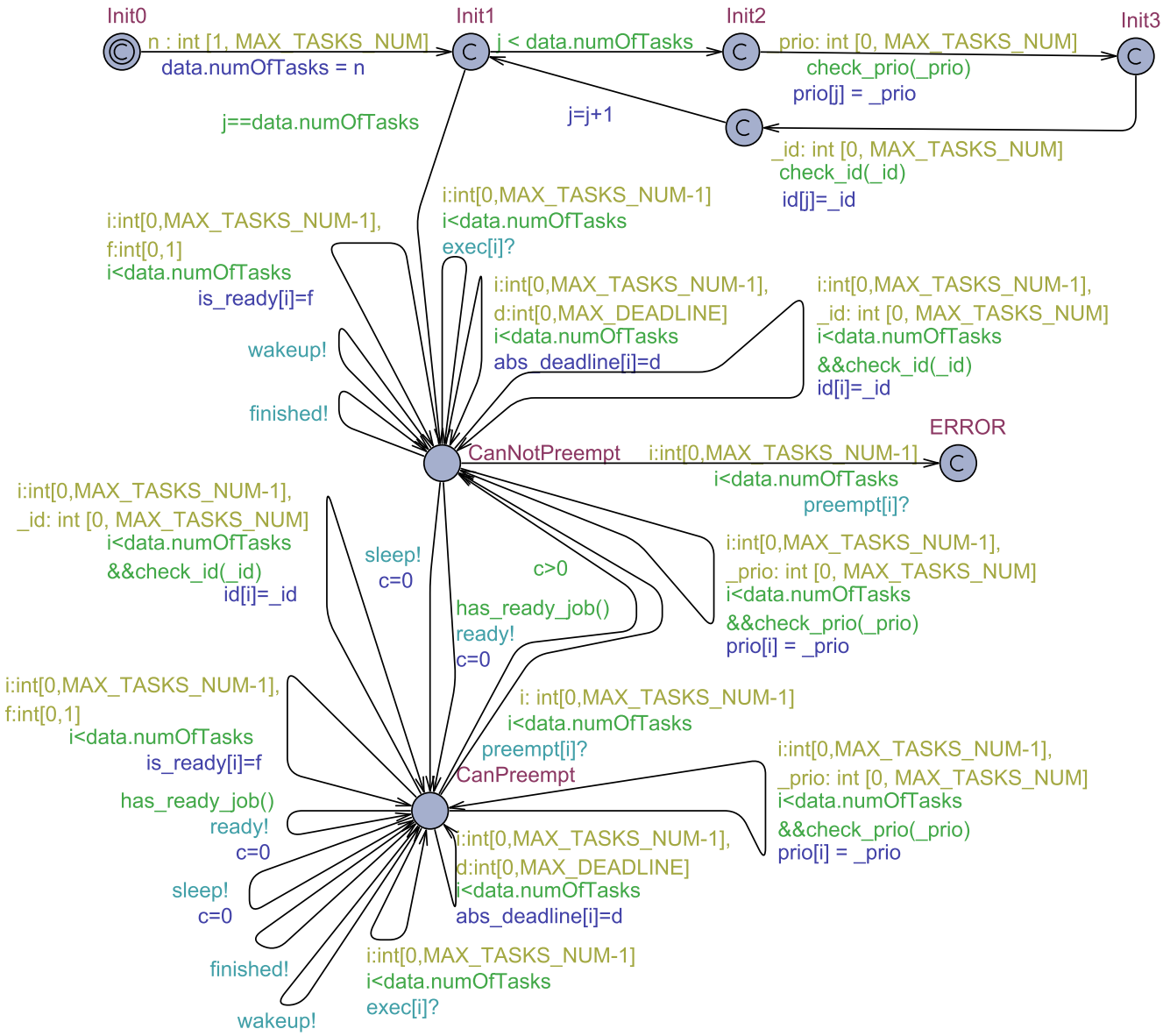

Рисунок В.8 — Автомат-наблюдатель, соответствующий требованию 5 к модели планировщика раздела

Требование 8. *Для любого раздела верно, что планировщик этого раздела мгновенно реагирует на события открытия и закрытия очередного окна этого раздела* [119].

В терминах синхронизаций автоматов это требование может быть сформулировано следующим образом: как только в автомате-модели функциональной задачи, функционирующем совместно с автоматом-моделью планировщика соответствующего раздела, становится активным переход с синхронизацией по каналу $wakeup$, либо по каналу $sleep$, эта синхронизация мгновенно выполняется.

Для того, чтобы требование не выполнялось, необходимо и достаточно, чтобы существовало такое вычисление сети автоматов (состоящей из исходного автомата и автомата, функционирующего с ним совместно), что после того, как во внешнем автомате становится активным переход с синхронизацией по каналу $wakeup$ или $sleep$, синхронизации по этим каналам не происходит в течение ненулевого времени.

Автомат-наблюдатель, соответствующий требованию 8, приведен на рисунке В.9. Помимо локаций $\{Init0, ..., Init3\}$, в автомате-наблюдателе присутствуют локации $L0$, $L1$ и «плохая» локация $ERROR$. Когда текущей локацией является локация $L0$, в автомате-наблюдателе активны переходы с синхронизациями по всем каналам, кроме $wakeup$ и $sleep$. Из локации $L0$ имеется переход в локацию $L1$, который может быть выполнен в любой момент времени. При выполнении этого перехода обнуляется значение таймера c. Когда текущей локацией является локация $L0$, в автомате-наблюдателе активны переходы с синхронизациями по всем каналам, в том числе $wakeup$ и $sleep$. Переходы с синхронизациями по каналам $wakeup$ и $sleep$ ведут в локацию $L0$. Если локация $L1$ является текущей в течение ненулевого времени, то это означает, что переход с синхронизацией по каналу $wakeup$ или $sleep$ активен и не выполняется в течение ненулевого времени, следовательно, требование 8 нарушается. Поэтому, если значение таймера c больше нуля, то из локации $L1$ выполняется переход в локацию $ERROR$. Аналогично предыдущим автоматам-наблюдателям, в описанном автомате-наблюдателе имеются переходы с недетерминированным изменением переменных is_ready, $abs\_deadline$, $prio$, id. Таким образом, этот автомат-наблюдатель позволяет смоделировать появление активных переходов с синхронизациями по каналам $wakeup$ и $sleep$ в любой момент времени, при всех возможных значениях переменных, и проверить, что эти переходы выполняются мгновенно после их появления.

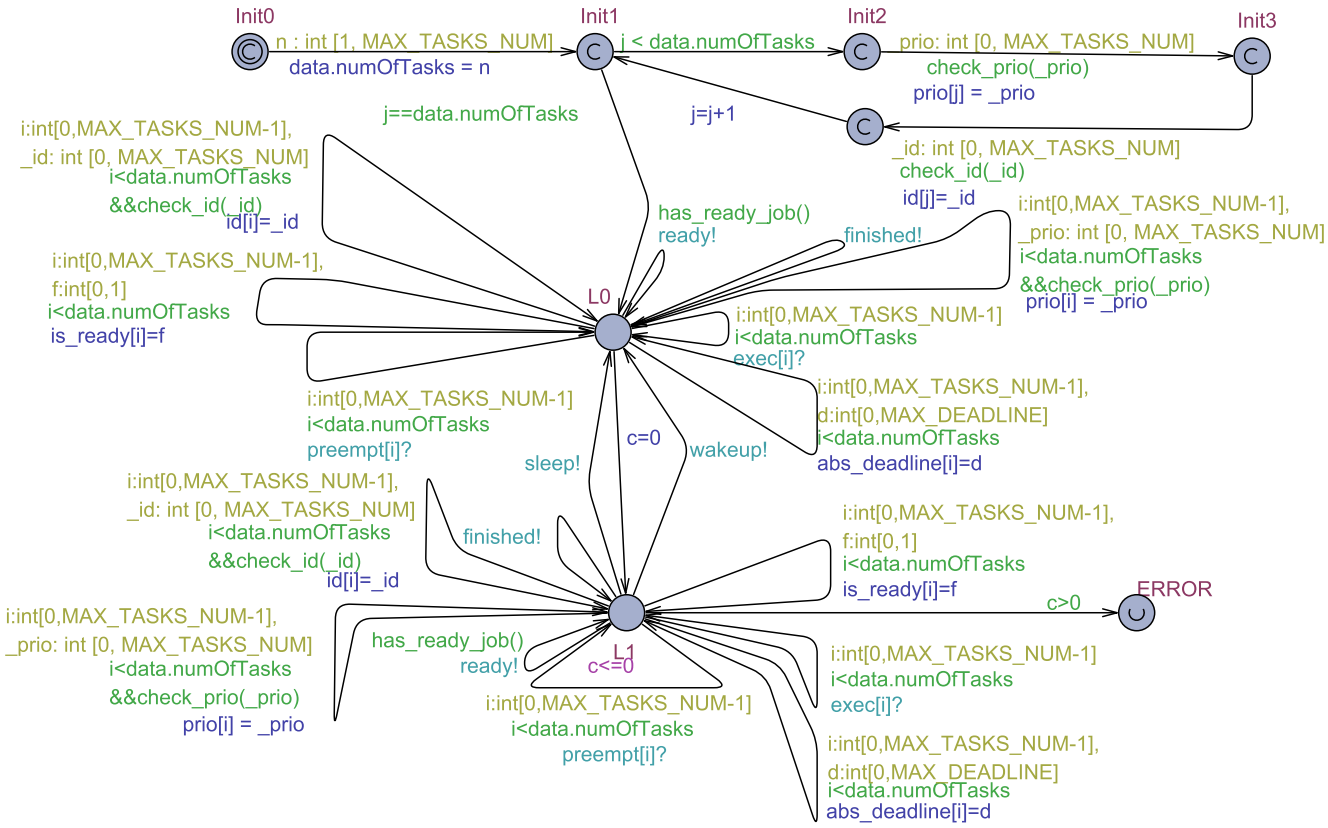

Рисунок В.9 — Автомат-наблюдатель, соответствующий требованию 8 к модели планировщика раздела

B.2. Автоматы для требований к моделям функциональных задач

Требование 1. *Для любой задачи верно, что некоторая ее работа может быть поставлена на выполнение, только если эта работа является готовой* [45].

Согласно разделу 2.4, готовности работы i-й задачи соответствует значение 1 переменной is_ready_i, а для оповещения модели планировщика о готовности работы некоторой задачи используется синхронизация по каналу $ready$. Постановка на выполнение текущей работы i-й задачи моделируется синхронизацией по каналу $exec_i$. Согласно разделу 2.5, модель i-й задачи имеет доступ только к i-й переменной массива is_ready и только к i-му каналу массива $exec$, поэтому для краткости в данном разделе is_ready_i будем обозначать как is_ready, а $exec_i$ — как $exec$.

В терминах обобщенной модели функционирования МВС (далее для краткости будем использовать формулировку «в терминах обобщенной модели») требование 1 может быть переформулировано следующим образом: *Для любой задачи верно, что некоторая ее работа может быть поставлена на выполнение* (*может произойти синхронизация по каналу* $exec$), *только если она является готовой* (*переменная* is_ready *равна* 1) *и планировщик соответствующего раздела оповещен о ее готовности* (*выполнена синхронизация по каналу* $ready$). *При этом, если планировщик оповещается о готовности очередной работы раздела* (*выполняется синхронизация по каналу* $ready$), *то работа этой задачи действительно является готовой* (*переменная* is_ready *равна* 1). Фиксированная работа может стать готовой лишь один раз за время своего существования в системе (i-я работа задачи с периодом $period$ существует в системе в течение интервала $[period \cdot (i-1), period \cdot i]$), поэтому, если выполняющаяся работа была вытеснена, то для ее повторной постановки на выполнение повторного оповещения о готовности не требуется (но это оповещение *может* быть выполнено).

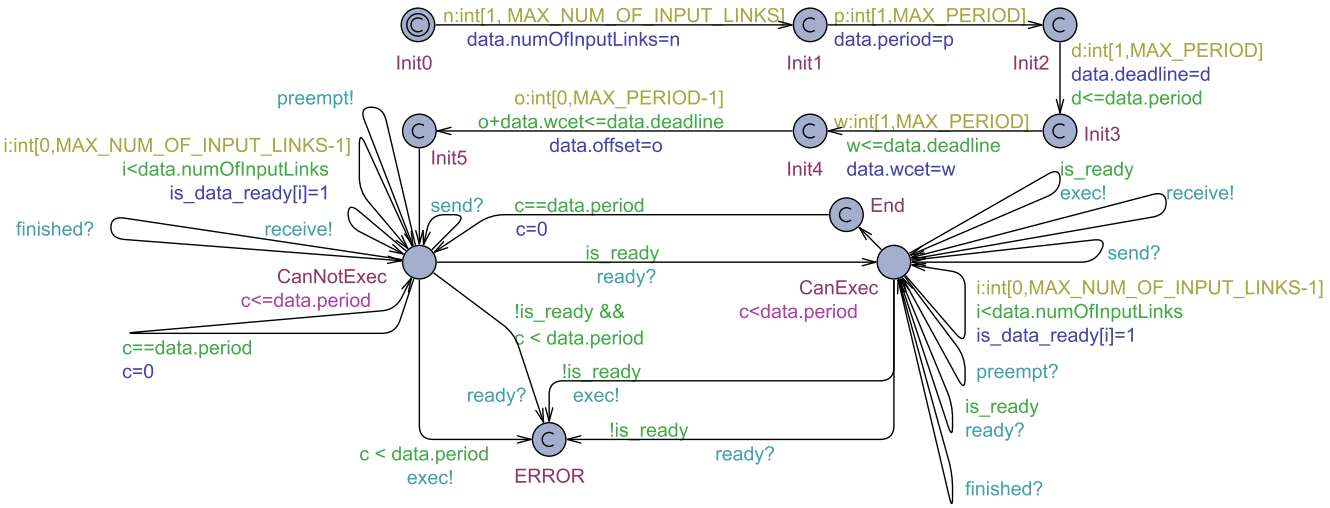

Рисунок B.10 — Автомат-наблюдатель, соответствующий требованию 1 к модели функциональной задачи

Автомат-наблюдатель, соответствующий требованию 1, приведен на рисунке B.10. При переходах между приоритетными локациями $\{Init0, ..., Init5\}$ происходит недетерминированная

установка значений параметров автомата: выбирается количество задач, от которых моделируемая задача зависит по данным, выбирается период, правая и левая относительные границы директивных интервалов, максимальное время выполнения на вычислителе (выбор диапазонов значений параметров описан далее). Обеспечивается корректность перечисленных параметров: правая относительная граница директивных интервалов должна быть больше левой и обе границы не должны превышать период; левая относительная граница директивных интервалов и максимальное время выполнения в сумме не должны превышать правую границу. Описанная схема установки значений параметров является одинаковой для всех автоматов-наблюдателей, используемых для проверки выполнения требований к моделям функциональных задач.

Помимо локаций $\{Init0, ..., Init5\}$, в автомате-наблюдателе присутствуют локации $CanNotExec$, $CanExec$ и End, соответствующие состояниям функциональной задачи до готовности очередной работы, готовности работы и окончания существования работы, а также «плохая» локация $ERROR$. Интерфейс базового типа T содержит набор переменных is_data_ready. С помощью присваивания этим переменным значения 1 в автоматах, реализующих базовый тип L, моделируется окончание передачи очередного сообщения. С помощью присваивания этим переменным значения 0 в автоматах, реализующих базовый тип T, моделируется считывание данных очередного сообщения. Поэтому для каждой из локаций $CanNotExec$ и $CanExec$ имеется переход в ту же локацию с присваиванием этим переменным значения 1 (значение 0 присваивается только изнутри автомата, реализующего TS, а значение 1 — извне, поэтому в автомате-наблюдателе присваивается только значение 1). Локация End — «служебная» и используется только для обнуления значения таймера c, необходимого для измерения времени с начала очередного периода.

Покажем, как выбираются диапазоны значений параметров в соответствии с приложением Б на примере разработанной автором модели функциональной задачи (описание модели см. в разделе А.3 приложения А). Модель работает с одним массивом и удовлетворяет условиям 1—6 и 8—11 определения класса автоматов У6 на стр. 165. Автомат-наблюдатель также удовлетворяет условиям 1—6 и 8—11 определения класса автоматов У6. Условие 7 определения класса автоматов У6 выполняется по построению автомата-наблюдателя. Поэтому, согласно рассуждениям на стр. 177, достаточные для верификации значения параметра-размера массива определяются в соответствии с утверждением 6. Суммарное количество функций и индексных переменных в модели равно 1. В автомате-наблюдателе функций и индексных переменных нет. Поэтому, согласно утверждению 6, достаточно провести верификацию для массивов размеров, не превышающих 1. Для модели и автомата-наблюдателя выполняются достаточные условия периодичности, сформулированные на стр. 162, и оба автомата удовлетворяет условиям утверждения 11. Количество временных параметров равно 4, суммарное количество таймеров в модели и в автомате-наблюдателе равно 2 (один из них — с возможностью останова). Таким образом, согласно утверждению 11, в качестве значений временных параметров достаточно рассматривать числа, не превышающие 22. В модели используются только булевы переменные, поэтому для них перебираются все возможные значения. Индексные параметры в модели не используются.

Выбор значений параметров и переменных при проверке остальных требований к модели функциональной задачи осуществляется аналогично: все автоматы-наблюдатели, соответствующие требованиям, удовлетворяют достаточным условиям периодичности и условиям утверждения 11, и имеют не более одного таймера с возможностью останова.

Требование 2. *Для любой задачи верно, что очередная ее работа становится готовой мгновенно после того как становится истинной конъюнкция трех условий: (1) модельное время не меньше левой границы директивного интервала; (2) получены синхронные сообщения от всех соответствующих работ-отправителей (если задача зависит по данным от других задач), (3) модельное время меньше правой границы директивного интервала.*

Согласно разделу 2.4, доставке i-го синхронного сообщения соответствует значение 1 переменной $is\_data\_ready_i$, а для оповещения модели задачи о доставке некоторого сообщения используется синхронизация по каналу $receive$. При этом сам факт синхронизации по каналу $receive$ несуществен для выполнения условия (2), существенно равенство 1 соответствующих переменных is_data_ready. Количество входных синхронных сообщений, относительные левая и правая границы директивных интервалов работ, период задачи обозначены соответственно как как $numOfInputLinks$, $offset$, $deadline$ (не путать с $abs\_deadline$ — переменной, содержащей значение абсолютной правой границы директивного интервала очередной работы задачи), $period$. Далее для краткости под утверждением «все переменные is_data_ready равны x» понимается «при наличии входных синхронных сообщений все переменные is_data_ready равны x». Аналогично, «хотя бы одна из переменных is_data_ready равна y» — «при наличии входных синхронных сообщений хотя бы одна из переменных is_data_ready равна y». Синхронные зависимости по данным могут существовать только между задачами с одинаковыми периодами и длительность выполнения работы задачи на вычислителе не может быть равна нулю. Поэтому, если в момент времени, равный $(k-1) \cdot p$ (k — некоторое натуральное число, p — период задачи) моделируется получение сообщения, то считается, что это сообщение предназначается k-й работе, а не $(k+1)$-й, так как ни одна из работ-отправителей для $(k+1)$-й работы не могла успеть выполниться за 0 единиц времени с момента начала периода.

Для начала приведем автомат-наблюдатель для проверки вспомогательного требования: для любой задачи, имеющей входные сообщения, верно, что очередная ее работа, соответствующая некоторому периоду, может использовать лишь те сообщения, которые получены от работ, соответствующих тому же самому периоду. В терминах обобщенной модели это означает, что в начале очередного периода все переменные is_data_ready должны быть равны 0.

Автомат-наблюдатель для проверки выполнения этого вспомогательного требования приведен на рисунке В.11. Помимо локаций $\{Init0, ..., Init5\}$, аналогичных локациям $\{Init0, ..., Init5\}$ предыдущего автомата-наблюдателя, автомат-наблюдатель для проверки выполнения рассматриваемого требования имеет локации $Work$ (соответствует функционированию очередной работы в течение соответствующего периода), $WaitForReset$ (соответствует ожиданию обнуления переменных is_data_ready в конце периода), $ERROR$ («плохая» локация) и вспомогательные локации $L1$ и $L2$. Аналогично автомату-наблюдателю для проверки требования 1, в автомате-наблюдателе для проверки рассматриваемого вспомогательного требования имеется переход из локации $Work$ в нее же с присваиванием переменным is_data_ready значения 1.

В отличие от автомата-наблюдателя для проверки предыдущего требования, для описываемого автомата существенно, что оповещение модели задачи о доставке синхронного сообщения (синхронизация по каналу $receive$) происходит мгновенно после окончания передачи сообщения (присваивания соответствующей переменной is_data_ready значения 1). Поэтому в автомате-наблюдателе сразу после присваивания переменной is_data_ready значения 1 происходит синхронизация по каналу $receive$. Требование мгновенности этого оповещения соответствует выполнению требования 2 к моделям виртуальных каналов.

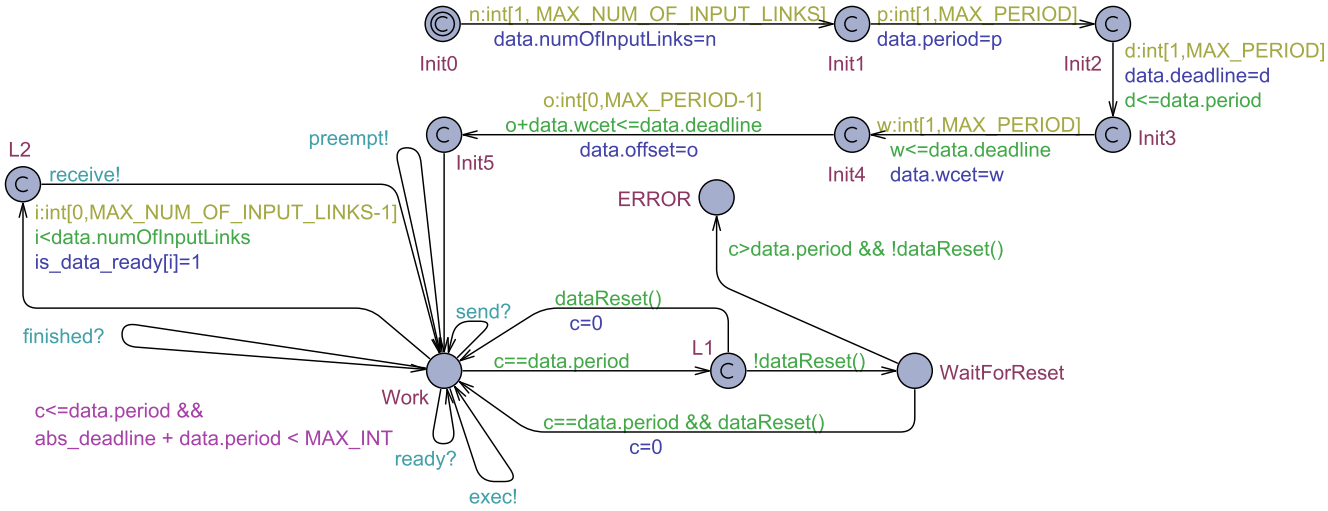

Рисунок В.11 — Автомат-наблюдатель, соответствующий вспомогательному требованию для требования 2 к модели функциональной задачи

Теперь опишем непосредственно автомат-наблюдатель для проверки выполнения требования 2, приведенный на рисунке В.12. Помимо локаций $\{Init0, ..., Init5\}$, аналогичных локациям $\{Init0, ..., Init5\}$ автомата-наблюдателя для проверки выполнения требования 1, автомат-наблюдатель для проверки выполнения требования 2 имеет локации $NotReady1$ (соответствует состоянию очередной работы до наступления левой границы ее директивного интервала), $NotReady2$ (соответствует состоянию очередной работы после наступления левой границы ее директивного интервала и до получения всех необходимых синхронных сообщений), $Ready$ (соответствует готовности очередной работы), $WaitForNewPeriod$ (соответствует состоянию очередной работы после уведомления планировщика о ее готовности и до завершения очередного периода) и $ERROR$ («плохая» локация). Аналогично автомату-наблюдателю для проверки требования 1, в автомате-наблюдателе для проверки требования 2 имеются переходы из каждой из локаций $NotReady1$, $NotReady2$, $Ready$ и $WaitForNewPeriod$ в ту же локацию с присваиванием переменным is_data_ready значения 1.

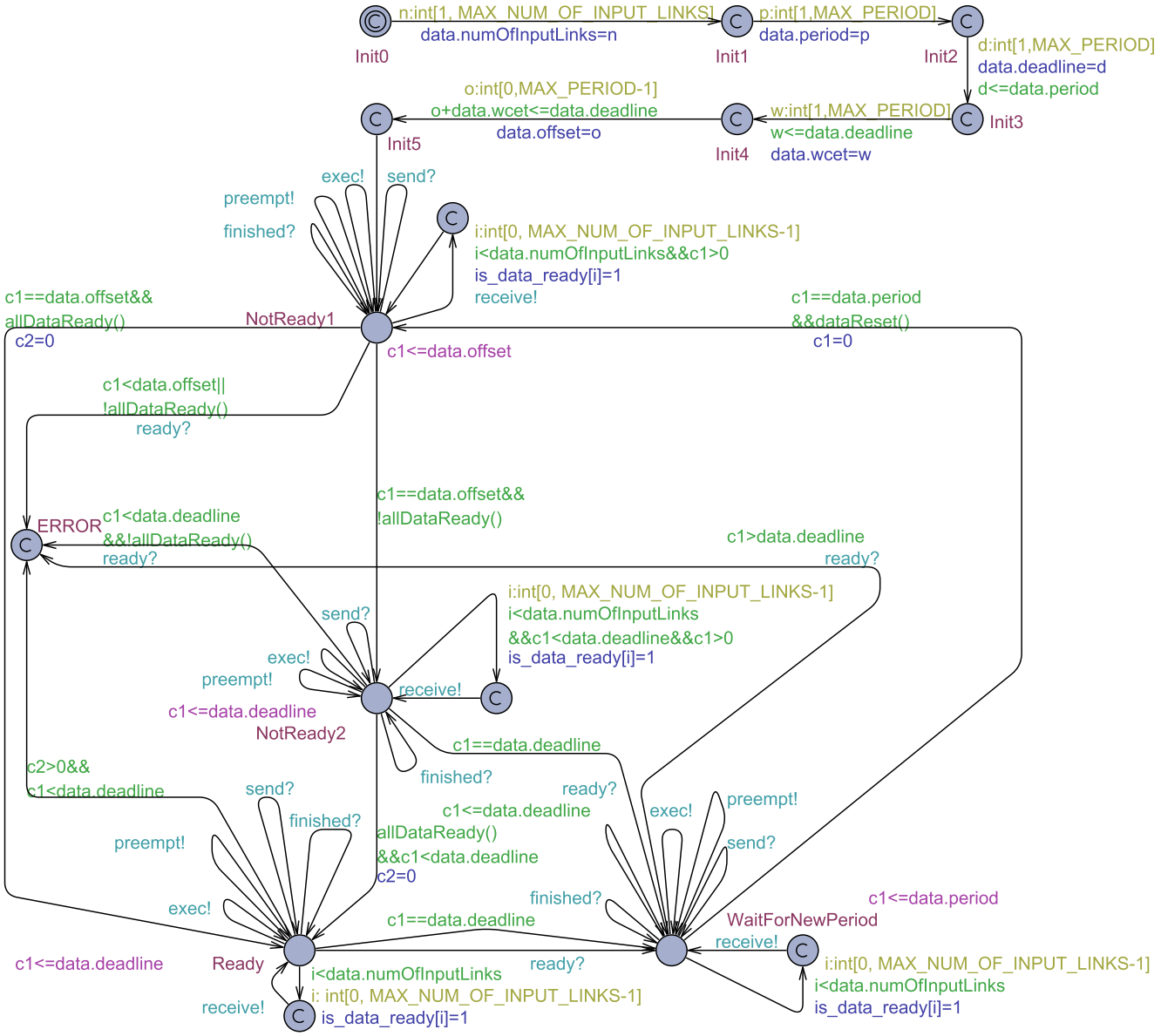

Рисунок В.12 — Автомат-наблюдатель, соответствующий требованию 2 к модели функциональной задачи

Требование 3. *Выполняющаяся работа задачи должна завершиться, как только станет выполненным хотя бы одно из следующих условий: длительность ее выполнения на вычислительном ядре достигло значения WCET; текущее время достигло правой границы директивного интервала этой работы. Работа не может завершиться, если ни одно из двух указанных условий не выполнено. Работа также не может завершиться, если она не является выполняющейся.*

В терминах обобщенной модели завершению выполняющейся работы задачи соответствует синхронизация по каналу $finished$, относительная правая граница директивного интервала обозначается как $deadline$.

Момент времени, когда суммарная длительность интервалов нахождения работы на вычислителе достигло максимальной возможной длительности выполнения работы, обозначим $JOB\_COMPLETE$. Это момент времени, когда суммарная длительность интервалов времени между последовательными синхронизациями по каналам $exec$ и $preempt$ плюс длительность интервала времени с момента последней синхронизации по каналу $exec$ достигло значения WCET.

Если в некоторый момент времени для текущей работы суммарная длительность интервалов времени между последовательными синхронизациями по каналам $exec$ и $preempt$ плюс длительность интервала времени с момента последней синхронизации по каналу $exec$ строго меньше WCET, то считается, что момент $JOB\_COMPLETE$ не наступил.

Автомат-наблюдатель для проверки выполнения требования 3 приведен на рисунке В.13. Помимо локаций $\{Init0, ..., Init5\}$, аналогичным локациям $\{Init0, ..., Init5\}$ предыдущих автоматов-наблюдателей, автомат-наблюдатель для проверки выполнения требования 3 имеет локации $Wait$ (соответствует состоянию очередной работы, в котором она *не* выполняется на вычислителе), $Exec$ (соответствует состоянию выполнения очередной работы на вычислителе), $Finish$ (соответствует завершению выполняющейся работы), $WaitForNewPeriod$ (соответствует ожиданию наступления очередного периода задачи) и $ERROR$ («плохая» локация). Аналогично предыдущим автоматам-наблюдателям, в автомате-наблюдателе для проверки требования 3 имеются переходы из каждой из локаций $Exec$, $Wait$, $Finish$ и $WaitForNewPeriod$ в ту же локацию с присваиванием переменным is_data_ready значения 1. Для измерения суммарной длительности интервалов выполнения работы используется таймер $c1$, для измерения времени с момента начала очередного периода — $c2$. Отметим, что работа не может быть поставлена на выполнение спустя $period$ единиц времени после начала своего периода (так как в этот момент времени начинается период, соответствующей уже следующей работе). Поэтому в автомате-наблюдателе синхронизации по каналу $exec$ возможны только при значении $c2$ строго меньшем $period$.

Требование 3а. *Для любой работы любой задачи верно, что с момента ее завершения до момента окончания соответствующего периода задачи эта работа не может быть готовой.*

В терминах обобщенной модели требование может быть переформулировано следующим образом: для любой задачи и любого $i > 0$ верно, что если на временном интервале $[period \cdot (i-1),\ period \cdot i]$ происходит синхронизация по каналу $finished$, то с момента этой синхронизации до наступления момента $period \cdot i$ переменная is_ready не может быть равна 1.

Автомат-наблюдатель для проверки выполнения требования 3а приведен на рисунке В.14. Помимо локаций $\{Init0, ..., Init5\}$, аналогичных локациям $\{Init0, ..., Init5\}$ предыдущих автоматов-наблюдателей, автомат-наблюдатель для проверки выполнения требования 3а имеет локации $CanBeReady$ (соответствует состоянию очередной работы до ее завершения), $Finished$ (соответствует состоянию очередной работы после ее завершения) и $ERROR$ («плохая» локация). В части изменения переменных is_data_ready, описанный автомат-наблюдатель аналогичен автомату-наблюдателю, соответствующему требованию 3. Для измерения времени с момента начала очередного периода используется таймер c.

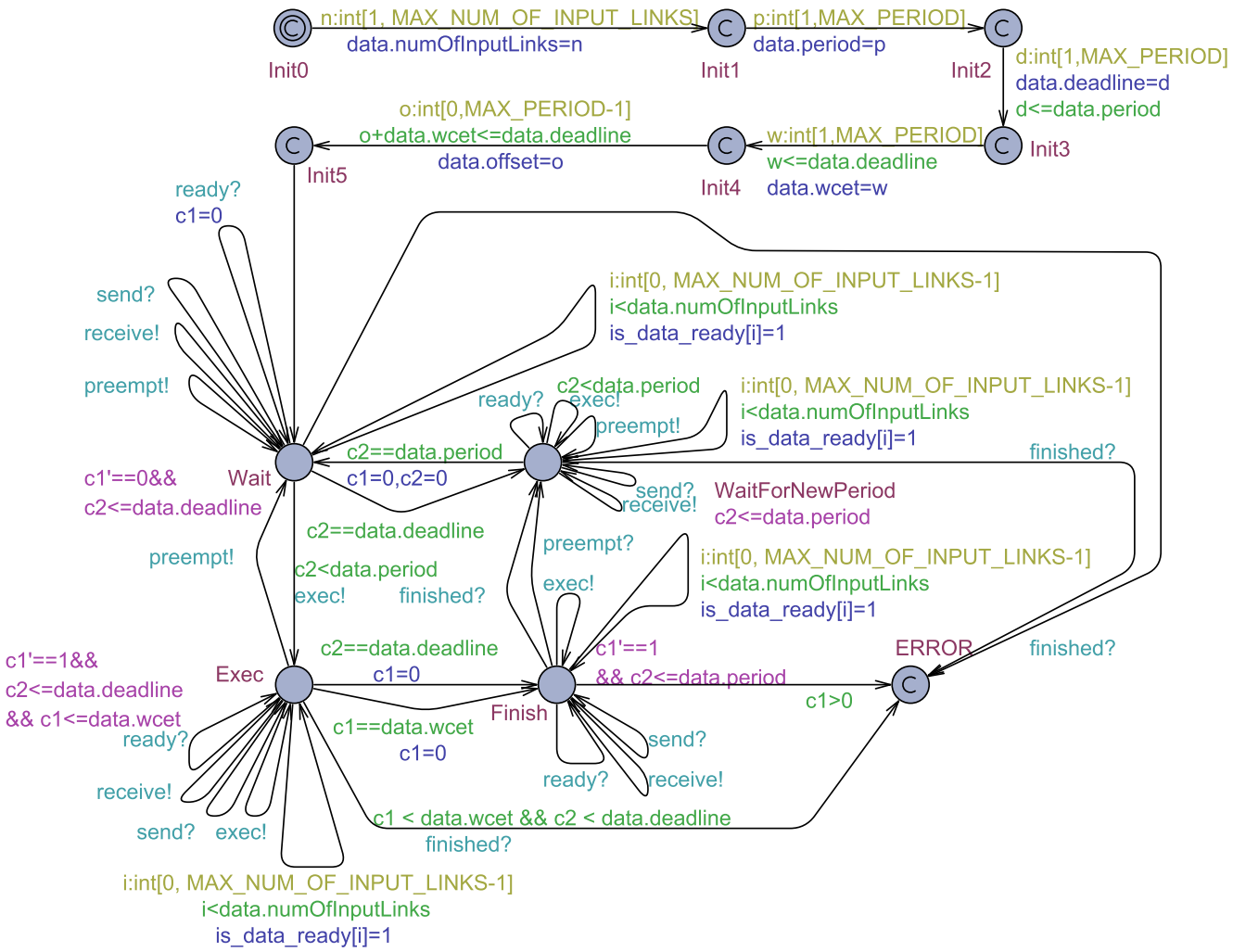

Рисунок В.13 — Автомат-наблюдатель, соответствующий требованию 3 к модели функциональной задачи

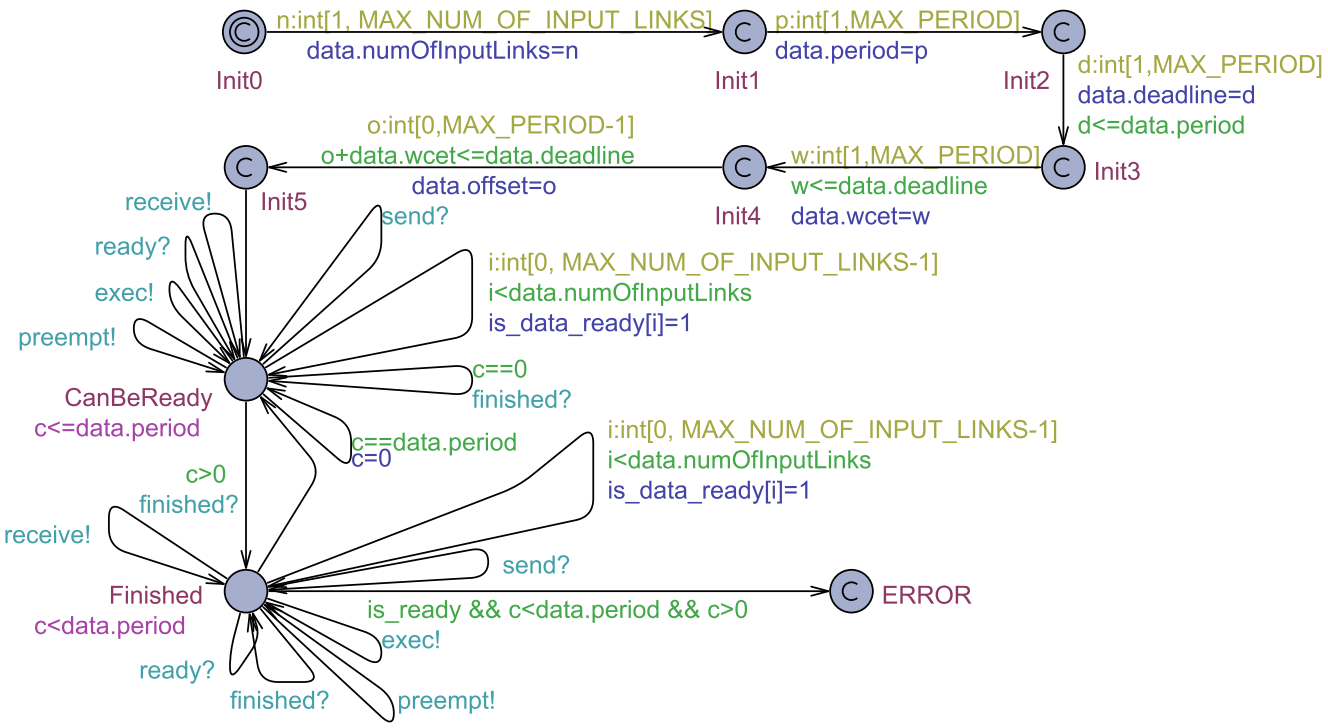

Рисунок В.14 — Автомат-наблюдатель, соответствующий требованию 3а к модели функциональной задачи

Требование 4. *Для любой задачи верно, что если эта задача имеет выходные сообщения, то каждая ее работа должна однократно послать сообщения через все выходные виртуальные каналы строго в момент завершения работы при условии штатного завершения. Если работа не успела штатно завершиться, то отправка сообщений не происходит.*

Согласно разделу 2.4, посылке работой сообщения соответствует синхронизация по каналу $send$. Этот канал является широковещательным и, согласно разделу 2.5, одной модели задачи соответствует один такой канал. Интерфейсы всех моделей виртуальных каналов, по которым должны передаваться сообщения от работ заданной задачи, содержат канал $send$, соответствующий этой задаче. Таким образом, при посылке моделью функциональной задачи сигнала по каналу $send$ модели всех виртуальных каналов, имеющих эту задачу в качестве отправителя, получают этот сигнал. Если таких виртуальных каналов нет (задача не имеет синхронных выходных зависимостей), то отправка сигнала по каналу $send$ происходит без синхронизации с другими автоматами. Следовательно, остается проверить, что при моделировании штатного завершения работы (в границах директивного интервала) в момент завершения (синхронизации по каналу $finished$) происходит однократная синхронизация по каналу $send$.

Автомат-наблюдатель для проверки выполнения требования 4 приведен на рисунке В.15. Помимо локаций $\{Init0, ..., Init5\}$, аналогичных локациям $\{Init0, ..., Init5\}$ предыдущих автоматов-наблюдателей, автомат-наблюдатель для проверки выполнения требования 4 имеет локации $Wait$ (соответствует состоянию очередной работы, в котором она *не* выполняется на вычислителе), $Exec$ (соответствует состоянию выполнения очередной работы на вычислителе), $Finish1$ (соответствует завершению работы, при этом уведомление планировщика о завершении выполняется перед посылкой сообщений), $Finish2$ (соответствует завершению работы, при этом посылка сообщений выполняется перед уведомлением планировщика о завершении), $WaitForNewPeriod$ (соответствует ожиданию наступления очередного периода задачи) и $ERROR$ («плохая» локация). В части использования таймеров $c1$ и $c2$, а также изменения переменных is_data_ready, описанный автомат-наблюдатель аналогичен автомату-наблюдателю, соответствующему требованию 3.

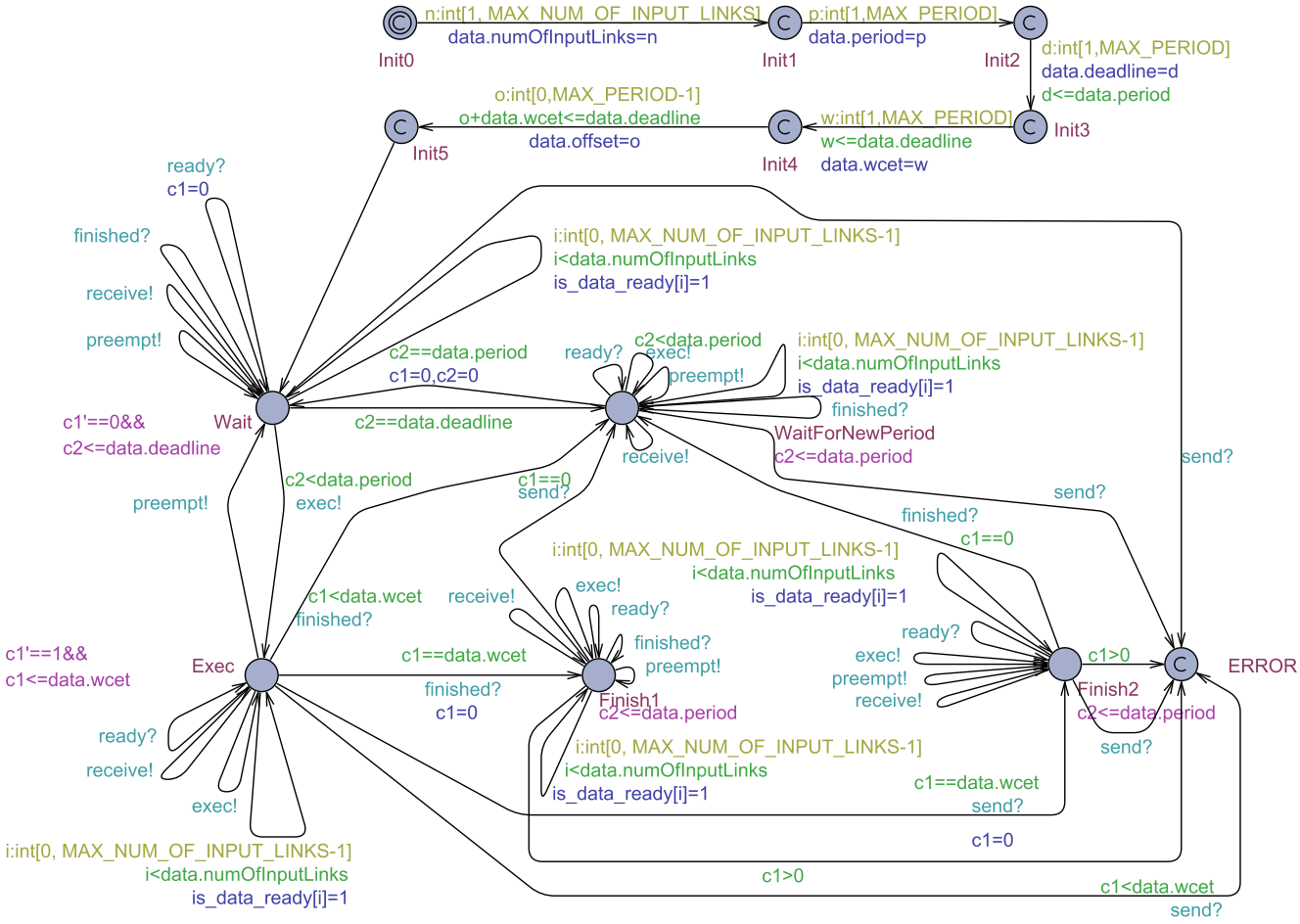

Рисунок В.15 — Автомат-наблюдатель, соответствующий требованию 4 к модели функциональной задачи

Требование 6. *Для любой задачи верно, что значение абсолютной правой границы директивного интервала ее очередной работы рассчитывается в соответствии с его определением, то есть значение абсолютной правой границы директивного интервала очередной работы равно времени начала соответствующего этой работе периода, увеличенному на значение относительной правой границы ее директивного интервала.*

Поскольку для модели задачи выполняются достаточные условия периодичности, сформулированные на стр. 162, то строго один раз в период $data.period$ выполняется обнуление таймера $local\_timer$, то есть действие $local\_timer = 0$. Следовательно, для того, чтобы проверить выполнение требования 6, достаточно проверить, что все переходы, содержащие действие $local\_timer = 0$, также содержат действие $abs\_deadline = abs\_deadline + data.period$ (возможно, записанное в виде $abs\_deadline \mathrel{+}= data.period$). Для разработанной модели функциональной задачи такая проверка была выполнена. Выполнение этой проверки, как и других аналогичных проверок, описанных в приложении Б, может быть автоматизировано.

B.3. Автоматы для требований к моделям планировщиков ядра

Согласно разделу 2.4, модель планировщика ядра посылает сигналы по каналам $wakeup_i$ и $sleep_i$, где i — номер раздела, привязанного к этому ядру. Синхронизации по этим каналам используются для уведомления моделей планировщиков разделов об открытии и закрытии очередного окна раздела. Согласно разделу 2.5, модель планировщика ядра по построению взаимодействует лишь с моделями планировщиков тех разделов, которые привязаны к этому ядру. Параметры автоматов, моделирующих планировщики ядра, задают расписание окон для ядра: длительность интервала планирования, число окон на интервале планирования, границы окон и соответствие окон разделам.

Требование 1. *Для любого ядра верно, что для любого окна раздела планировщик ядра однократно оповещает планировщик раздела, соответствующего данному окну, об открытии этого окна строго в момент, определенный в расписании.*

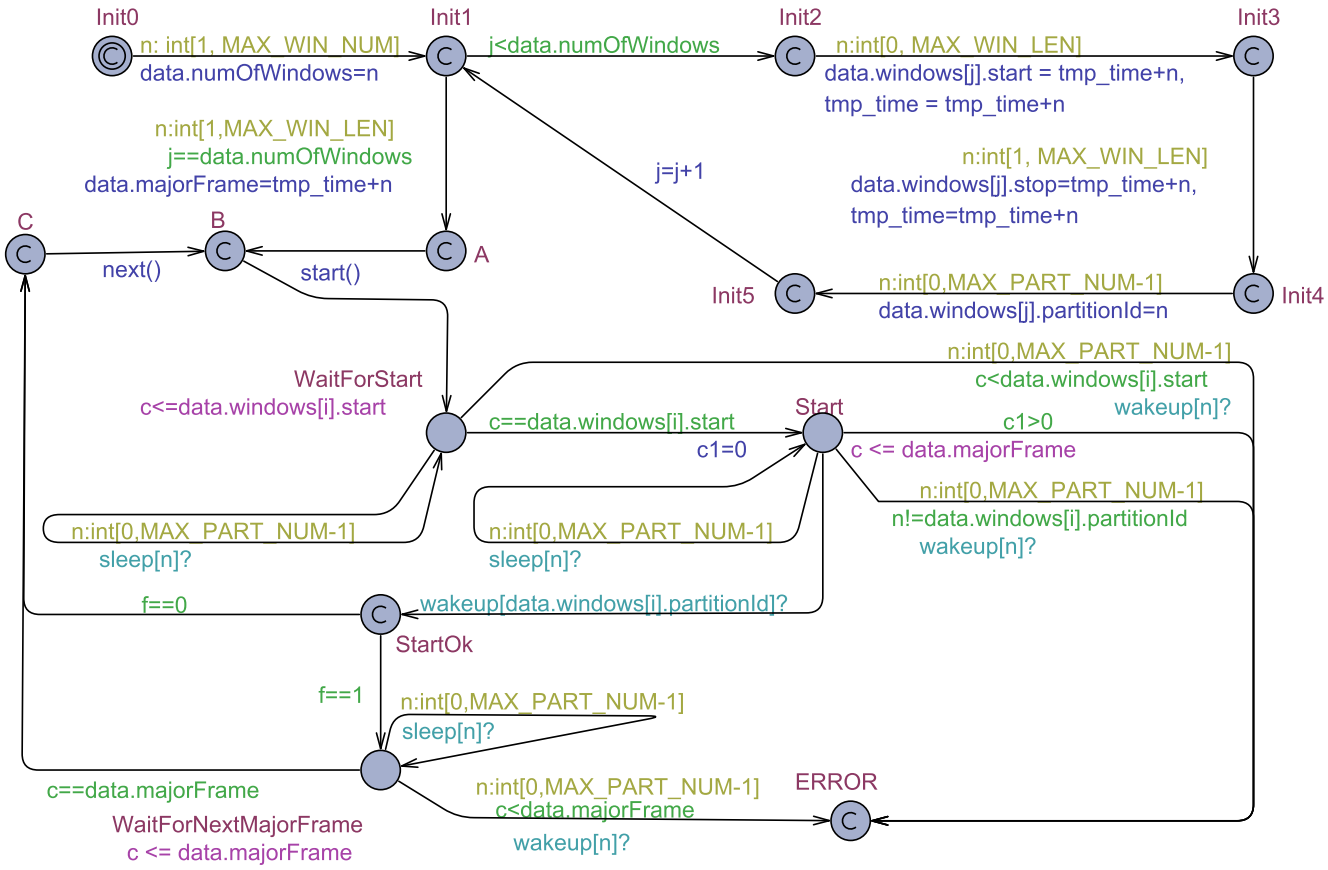

Рисунок B.16 — Автомат-наблюдатель, соответствующий требованию 1 к модели планировщика ядра

Автомат-наблюдатель для проверки выполнения требования 1 приведен на рисунке B.16. При переходах между локациями $\{Init0, ..., Init6\}$ происходит недетерминированная установка значений параметров автомата: выбирается количество окон на интервале планирования, затем для каждого окна выбирается время открытия, время закрытия и раздел, которому принадлежит окно (выбор диапазонов значений параметров описан далее). Параметры устанавливаются таким

образом, что обеспечивается выполнение ограничений корректности для расписания, введенных в разделе 1.4.1: окна не пересекаются, одно окно принадлежит одному разделу, время закрытия последнего окна относительно начала интервала планирования не превышает длительности интервала планирования.

Помимо локаций $\{Init0, ..., Init6\}$, в автомате-наблюдателе присутствуют локации $WaitForStart$ (ожидание открытия очередного окна), $Start$ (момент открытия очередного окна), $StartOk$ (корректное открытие очередного окна), $WaitForNextMajorFrame$ (ожидание завершения интервала планирования после открытия последнего окна), $ERROR$ («плохая» локация) и пара локаций A, B для перехода к следующему окну. Для измерения задержек между наступлением моментов открытия окна и синхронизациями по каналам $wakeup_i$ используется таймер $c1$; c — неостанавливаемый таймер, использующийся в сравнениях с временными параметрами.

Покажем, как выбираются диапазоны значений параметров в соответствии с приложением Б на примере разработанной автором модели планировщика ядра (описание модели см. в разделе А.1 приложения А). Модель не имеет переменных, работает с массивами и удовлетворяет условиям утверждения 7. Автомат-наблюдатель удовлетворяет условиям следствия утверждения 7, то есть содержит ограничения на синхронизации, выполняющиеся при обработке *одного* элемента массива, независимо от номера этого элемента (а именно на синхронизации по каналам $wakeup_i$). Поэтому, согласно утверждению 7 и следствию из него, достаточно провести верификацию для массивов размеров, не превышающих 3. Автомат-модель и автомат-наблюдатель являются периодическими и удовлетворяют условиям утверждения 10. Обоснование периодичности автомата-наблюдателя выполняется аналогично приведенному на стр. 129 обоснованию периодичности автомата-модели. При размерах массивов, не превышающих 3, количество временных параметров не превышает 6. Таймеров с возможностью останова ни в автомате-модели, ни в автомате-наблюдателе нет. Таким образом, согласно утверждению 10, в качестве значений временных параметров достаточно рассматривать числа, не превышающие 6. При размерах массивов, не превышающих 3, количество индексных параметров не превышает 3. Модель планировщика раздела и автомат-наблюдатель удовлетворяют условиям утверждения 8, поэтому в качестве значений индексных параметров достаточно рассматривать числа, не превышающие 3.

Требование 2. *Для любого ядра верно, что для любого окна раздела планировщик ядра однократно оповещает планировщик раздела, соответствующего данному окну, о закрытии этого окна строго в момент, определенный в расписании.*

Автомат-наблюдатель, соответствующий требованию 2, практически идентичен автомату-наблюдателю, соответствующему требованию 1, с тем лишь отличием, что на переходах между локациями синхронизации по каналам $wakeup_i$ (моделирующие открытие очередного окна раздела) заменены на синхронизации по каналам $sleep_i$ (моделирующие закрытие очередного окна раздела) и наоборот — синхронизации по каналам $sleep_i$ заменены на синхронизации по каналу $wakeup_i$; моменты открытия окон заменены на моменты их закрытия. Автомат-наблюдатель, соответствующий требованию 2, приведен на рисунке В.17.

Выбор диапазонов значений параметров при проверке выполнения требования 2 к модели планировщика ядра осуществляется аналогично выбору диапазонов значений параметров при

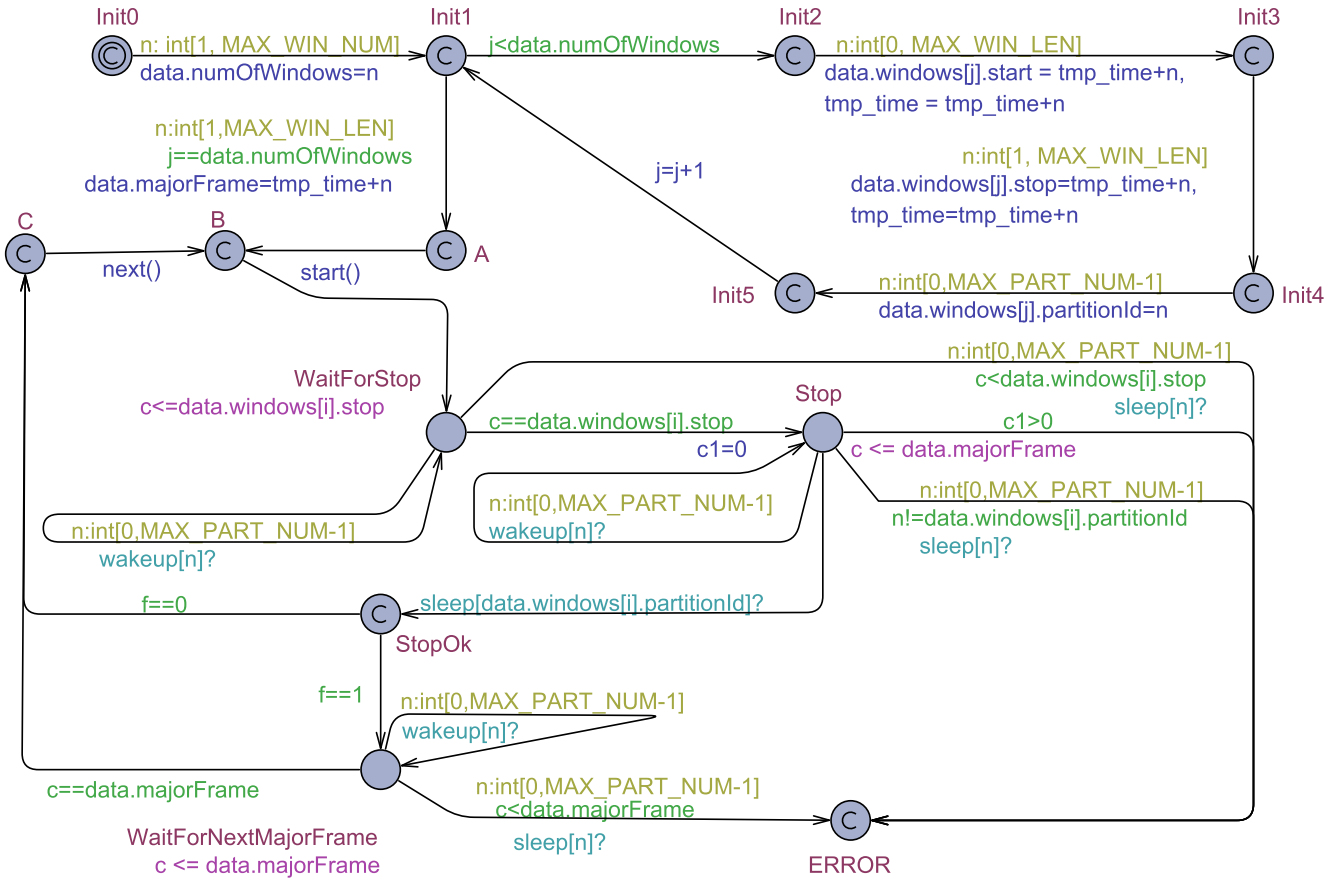

Рисунок В.17 — Автомат-наблюдатель, соответствующий требованию 2 к модели планировщика ядра

проверке выполнения требования 1: автомат-наблюдатель, соответствующий требованию 2, является периодическим (обоснование этого аналогично приведенному на стр. 129 обоснованию периодичности автомата-модели) и удовлетворяет условиям утверждений 8 и 10 и условиям следствия из утверждения 7.

В.4. Автоматы для требований к моделям виртуальных каналов

Согласно разделу 2.4, модель виртуального канала принимает сигналы по каналу $send$ (уведомление о начале передаче сообщения от модели функциональной задачи-отправителя) и отправляет сигналы по каналу $receive$ (уведомление о доставке сообщения для модели функциональной задачи-получателя). Согласно разделу 2.5, модель виртуального канала взаимодействует только с двумя моделями функциональных задач: посредством канала $send$ — с моделью задачи-отправителя, посредством канала $receive$ и переменной is_data_ready — с моделью задачи-получателя. Начало передачи сообщения моделируется синхронизацией по каналу $send$, окончание передачи — присваиванием переменной is_data_ready значения 1. Модель виртуального канала уведомляет модель задачи-получателя о доставке сообщения посредством синхронизации по каналу $receive$. Получению сообщения в модели задачи соответствует

присваивание переменной is_data_ready этой моделью значения 0. Параметр $delay$ автомата, моделирующего виртуальный канал, задает задержку на передачу сообщения. Согласно разделу 1.4.1, для каждого сообщения в конфигурации задается задержка на его передачу через память модуля и через сеть, а при построении модели конкретной МВС (раздел 2.5) для установки значения соответствующего параметра автомата выбирается одна из этих задержек в зависимости от привязки задачи-отправителя и задачи-получателя к ядрам.

Требование 1. *Для любого виртуального канала верно, что задержка на передачу сообщения через виртуальный канал строго равна значению, указанному в конфигурации системы.*

Задержкой на передачу сообщения считается интервал времени между синхронизацией по каналу $send$ и моментом, когда значение переменной is_data_ready становится равным 1.

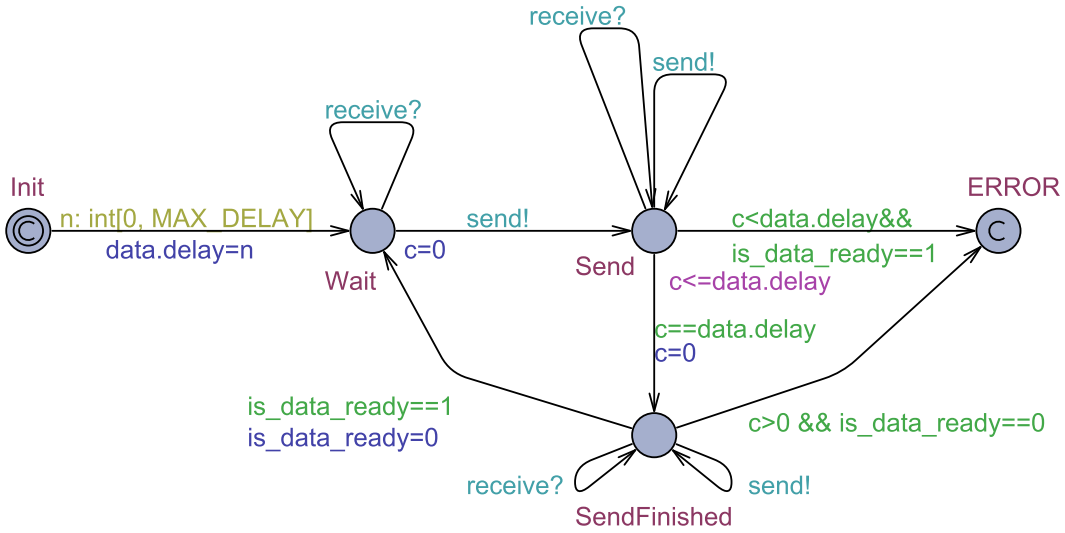

Рисунок В.18 — Автомат-наблюдатель, соответствующий требованию 1 к модели виртуального канала

Автомат-наблюдатель для проверки выполнения требования 1 приведен на рисунке В.18. При переходе из локации $Init$ в локацию $Wait$ происходит недетерминированная установка значения параметра автомата: выбирается задержка на передачу сообщения. Локация $Wait$ соответствует ожиданию начала передачи сообщения, $Send$ — передаче, $SendFinished$ — окончанию передачи, $ERROR$ — «плохая» локация. Для измерения задержек используется таймер c. После окончания передачи (присваивания переменной is_data_ready значения 1 в модели виртуального канала) в автомате-наблюдателе происходит присваивание переменной is_data_ready значения 0 (соответствует получению сообщения). Присваивание переменной is_data_ready значения 0 производится для проверки того, что модель виртуального канала устанавливает этой переменной значение 1 спустя ровно $delay$ единиц времени с момента синхронизации по каналу $send$.

Покажем, как выбираются диапазоны значений параметров в соответствии с приложением Б на примере разработанной автором модели виртуального канала (описание модели см. в разделе А.4 приложения А). Массивы, индексные параметры и целочисленные параметры в модели не используются. Модель не является периодической и имеет один временной параметр. Автомат-модель и автомат-наблюдатель удовлетворяют условиям утверждения 12, поэтому,

согласно утверждению 12, при верификации достаточно рассмотреть значения временного параметра, не превышающие 1.

Требование 2. *Для любого виртуального канала и любого сообщения, передаваемого по нему, верно, что получатель оповещается о получении сообщения ровно один раз, в момент окончания передачи.*

Согласно предыдущему требованию, окончание передачи происходит спустя ровно $delay$ единиц времени после начала передачи.

Автомат-наблюдатель для проверки выполнения требования 2 приведен на рисунке В.19. Локации этого автомата аналогичны соответствующим локациям предыдущего автомата-наблюдателя. Выбор диапазона значений параметра $delay$ при проверке выполнения требования 2 к модели виртуального канала осуществляется согласно утверждению 12 приложения Б (аналогично случаю требования 1).

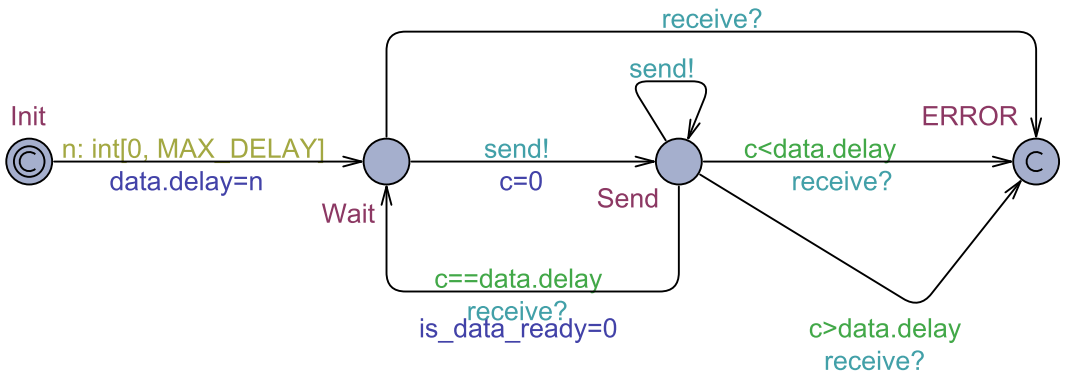

Рисунок В.19 — Автомат-наблюдатель, соответствующий требованию 2 к модели виртуального канала

В.5. Автоматы для требований к моделям конкретных типов компонентов МВС

В настоящем разделе приведены автоматы-наблюдатели, соответствующие требованиям к моделям различных планировщиков разделов.

Для всех автоматов-наблюдателей, описанных в настоящем разделе, выполняются условия, требующиеся для применения к этим автоматам-наблюдателям (совместно с соответствующими автоматами-моделями), утверждений 6 и 9 приложения Б. Поэтому выбор диапазонов значений параметров и переменных при верификации осуществляется согласно этим утверждениям. Пример применения этих утверждений к конкретной паре из автомата-модели планировщика раздела и автомата-наблюдателя, приведен на стр. 71.

B.5.1. Автоматы для требований к модели планировщика с фиксированными приоритетами и вытеснением

Требование 1. *Для любого раздела верно, что если выполнением работ этого раздела управляет планировщик с фиксированными приоритетами и вытеснением, то для постановки на выполнение всегда выбирается работа с максимальным среди всех готовых работ приоритетом*.

Автомат-наблюдатель, соответствующий требованию 1, приведен на рисунке В.20. Этот автомат аналогичен описанным выше автоматам-наблюдателям, соответствующим требованиям к моделям планировщиков. Помимо локаций $\{Init0, ..., Init3\}$ в автомате-наблюдателе присутствуют локации $Work$ (соответствует корректному функционированию планировщика) и $ERROR$ («плохая» локация).

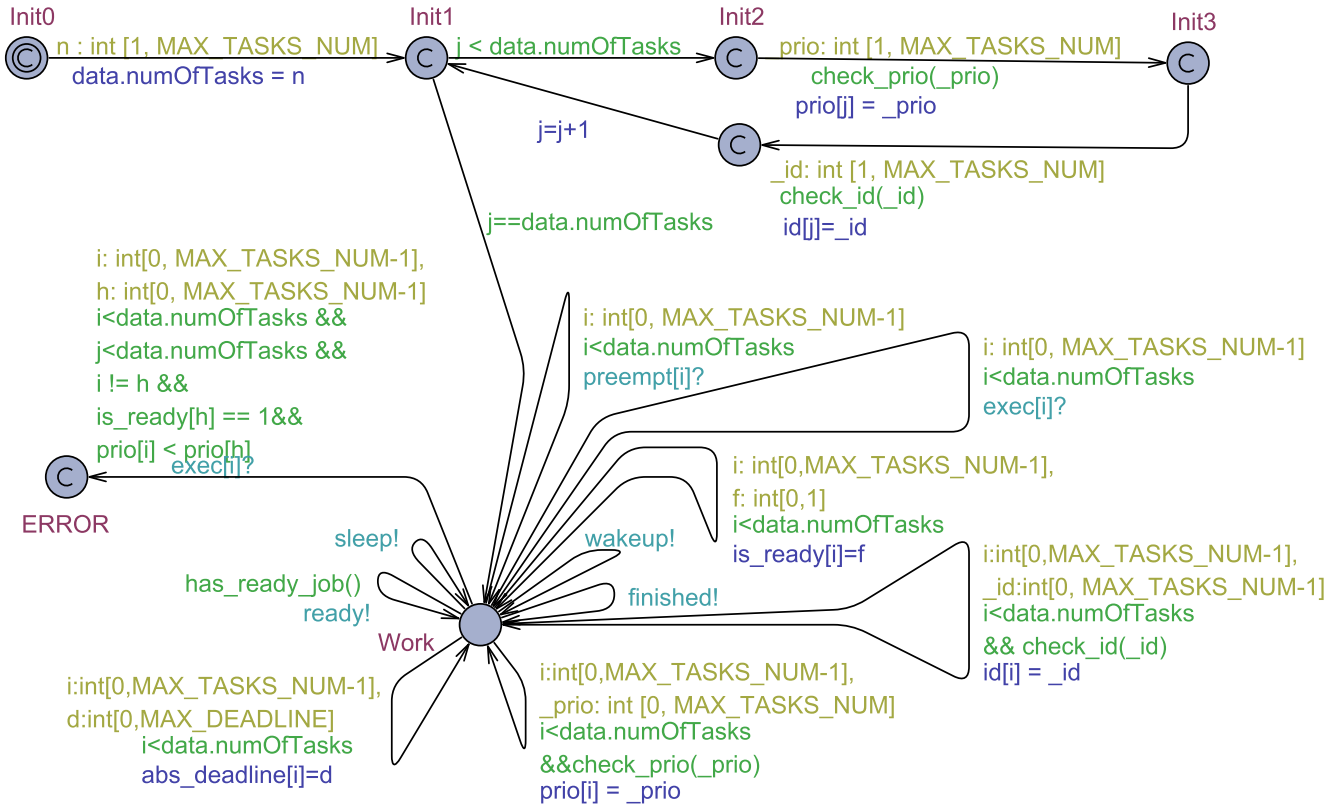

Рисунок В.20 — Автомат-наблюдатель, соответствующий требованию 1 к модели планировщика с фиксированными приоритетами и вытеснением

Требование 2. *Для любого раздела верно, что если выполнением работ этого раздела управляет планировщик с фиксированными приоритетами и вытеснением, то при появлении новой готовой работы с максимальным среди всех готовых работ приоритетом, некоторая работа мгновенно ставится на выполнение при условии того, что окно этого раздела открыто*.

Проверка того, что именно появившаяся готовая работа с максимальным среди всех готовых работ приоритетом ставится на выполнение, осуществляется при проверке требования 1.

Идея проверки требования 2 состоит в том, чтобы построить автомат-наблюдатель, моделирующий всевозможные внешние воздействия на модель планировщика раздела. При этом,

если воздействие соответствует появлению в рамках окна раздела новой готовой работы с максимальным среди всех готовых работ приоритетом, то реакция модели планировщика на такое воздействие должна соответствовать мгновенной постановке на выполнение некоторой работы (другая реакция должна приводить автомат-наблюдатель в «плохую» локацию).

Автомат-наблюдатель, соответствующий требованию 2, приведен на рисунке В.21. Локации $\{Init0, ..., Init3\}$ этого автомата аналогичны соответствующим локациям предыдущих автоматов-наблюдателей. Локация $Work$ соответствует корректному (по отношению к требованию 2) функционированию планировщика. Для каждого канала, входящего в интерфейс модели планировщика раздела, имеется переход из локации $Work$ в нее же с синхронизацией по этому каналу. Также имеются переходы из локации $Work$ в нее же с недетерминированным изменением переменных is_ready и $abs\_deadline$. Таким образом, при нахождении в локации $Work$ автомат-наблюдатель выполняет всевозможные воздействия на модель планировщика. При этом имеется переменная $part\_active$, которой присваивается значение 1 при синхронизации по каналу $wakeup$ и значение 0 при синхронизации по каналу $sleep$ (т.е. в рамках окон данного раздела переменная $part\_active$ равна 1). Также имеется переменная j, хранящая номер выполняющейся в данный момент работы и равная -1 при отсутствии выполняющихся работ. Этой переменной присваивается значение i при синхронизациях по каналу $exec_i$, и -1 при синхронизациях по каналам $preempt_i$ и $finished$. Для моделирования появления в рамках окна раздела новой готовой работы с максимальным приоритетом используются локации $L0$ и $L1$. Переход из локации $Work$ в локацию $L0$ возможен лишь при открытом окна раздела ($part\_active$ равна 1). При этом переходе выбирается работа, которая не выполняется в данный момент (ее номер не равен j) и приоритет которой выше, чем приоритеты прочих имеющихся на данный момент готовых работ (выбор такой работы реализован в функции get_job). Для выбранной работы моделируется готовность (соответствующей переменной is_ready присваивается значение 1 и выполняется синхронизация по каналу $ready$), и в этот момент обнуляется таймер c. Если спустя ненулевое время после этого выбранная работа не будет поставлена на выполнение и при этом окно раздела останется открытым, то произойдет переход в «плохую» локацию $ERROR$.

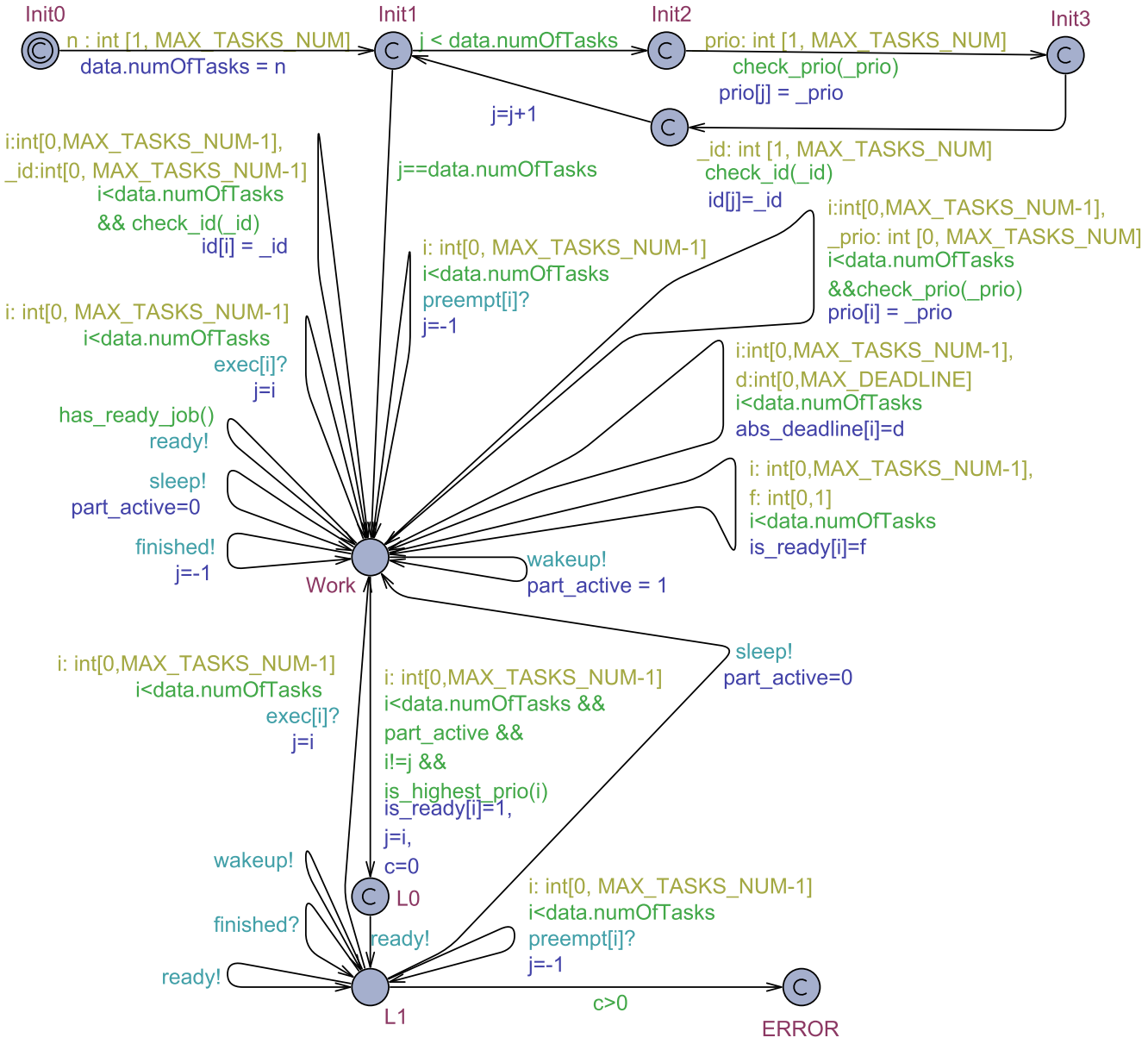

Рисунок В.21 — Автомат-наблюдатель, соответствующий требованию 2 к модели планировщика с фиксированными приоритетами и вытеснением

В.5.2. Автоматы для требований к модели планировщика, работающего по стратегии EDF с вытеснением

Требование 1. *Для любого раздела верно, что если выполнением работ этого раздела управляет планировщик, работающий по стратегии EDF с вытеснением, то для постановки на выполнение всегда выбирается готовая работа с наименьшей правой границей директивного интервала*.

Автомат-наблюдатель, соответствующий требованию 1, приведен на рисунке В.22. Этот автомат-наблюдатель аналогичен автомату-наблюдателю для проверки требования 1 к модели планировщика с фиксированными приоритетами и вытеснением. Отличие состоит лишь в том, что

переход в локацию $ERROR$ происходит в случае постановки на выполнение работы, правая граница директивного интервала которой не является наименьшей среди готовых работ.

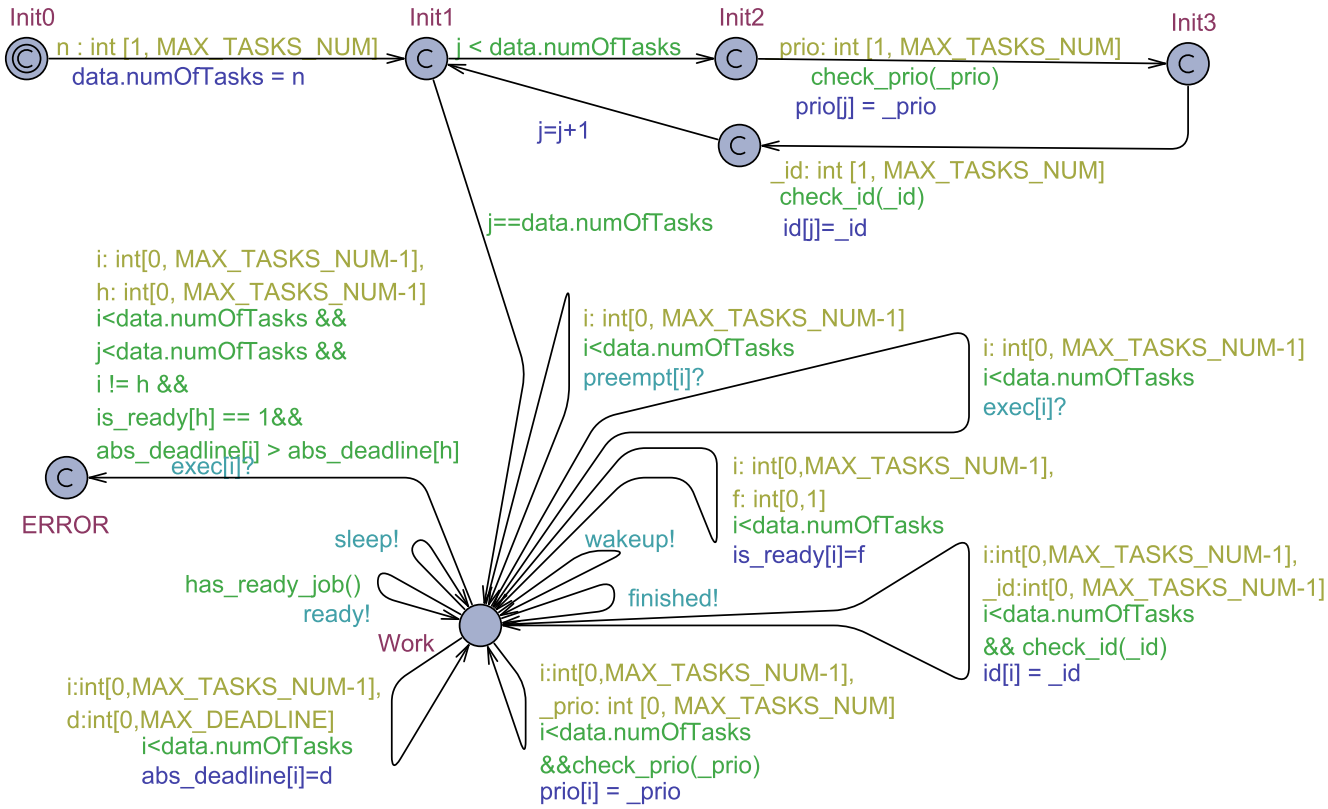

Рисунок В.22 — Автомат-наблюдатель, соответствующий требованию 1 к модели планировщика, работающего по стратегии EDF с вытеснением

Требование 1а. *Для любого раздела верно, что если выполнением работ этого раздела управляет планировщик, работающий по детерминированной стратегии EDF с вытеснением, то среди готовых работ с одинаковыми правыми границами директивных интервалов для постановки на выполнение выбирается та, номер задачи у которой меньше, чем у остальных*.

Автомат-наблюдатель для проверки требования 1а приведен на рисунке В.23. Этот автомат-наблюдатель аналогичен автомату-наблюдателю, соответствующему требованию 1, с тем лишь отличием, что переход в локацию $ERROR$ происходит в случае постановки на выполнение готовой работы, номер задачи которой больше, чем у прочих готовых работ с той же правой границей директивного интервала.

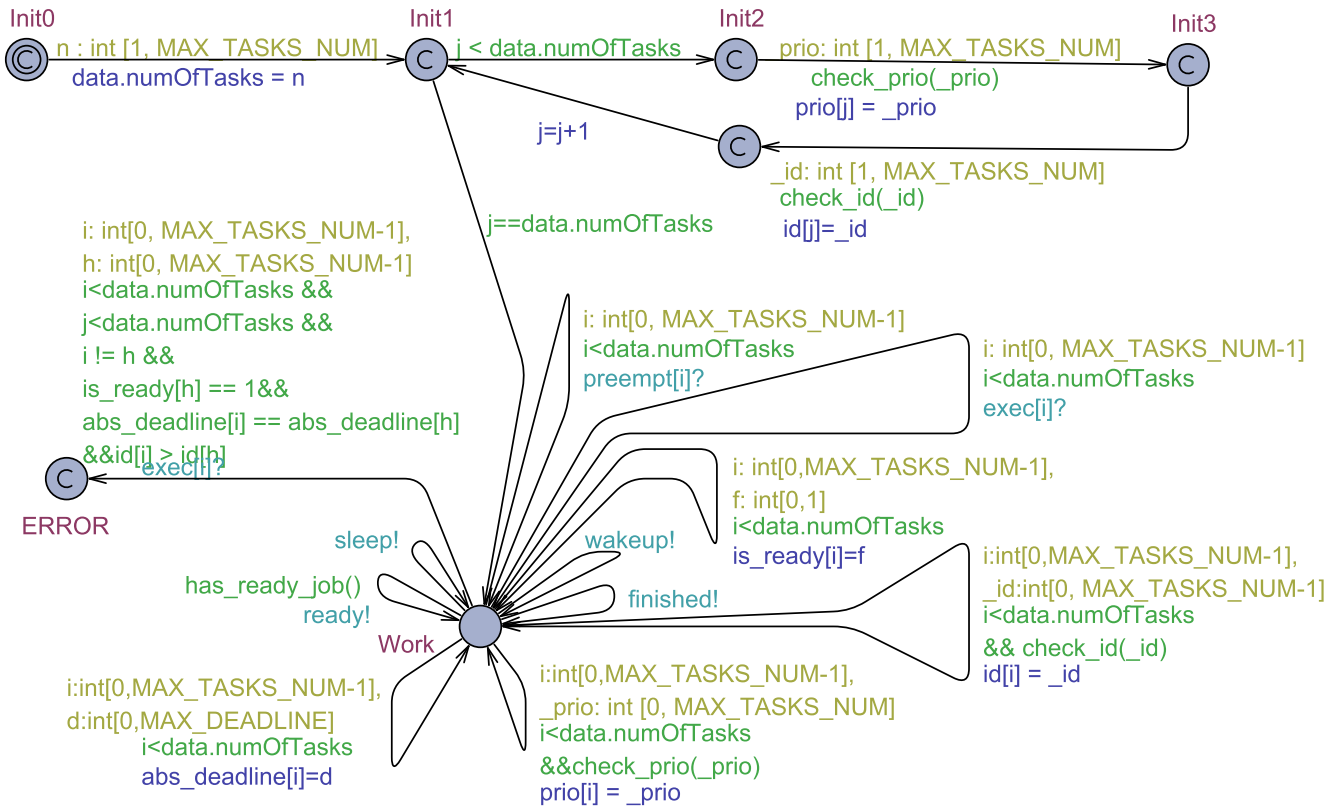

Рисунок В.23 — Автомат-наблюдатель, соответствующий требованию 1а к модели планировщика, работающего по стратегии EDF с вытеснением

Требование 2. *Для любого раздела верно, что если выполнением работ этого раздела управляет планировщик с фиксированными приоритетами без вытеснения, то при появлении новой готовой работы с меньшей, чем у всех готовых работ, правой границей директивного интервала, некоторая работа мгновенно ставится на выполнение при условии того, что окно этого раздела открыто*.

Автомат-наблюдатель, соответствующий требованию 2, приведен на рисунке В.24. Этот автомат-наблюдатель аналогичен автомату-наблюдателю для проверки требования 2 к модели планировщика с фиксированными приоритетами и вытеснением. Отличие состоит лишь в том, что при переходе из локации $Work$ в локацию $L0$ выбирается работа, правая граница директивного интервала которой меньше чем правые границы директивных интервалов прочих имеющихся на данный момент готовых работ. Этот выбор реализован в функции get_job.

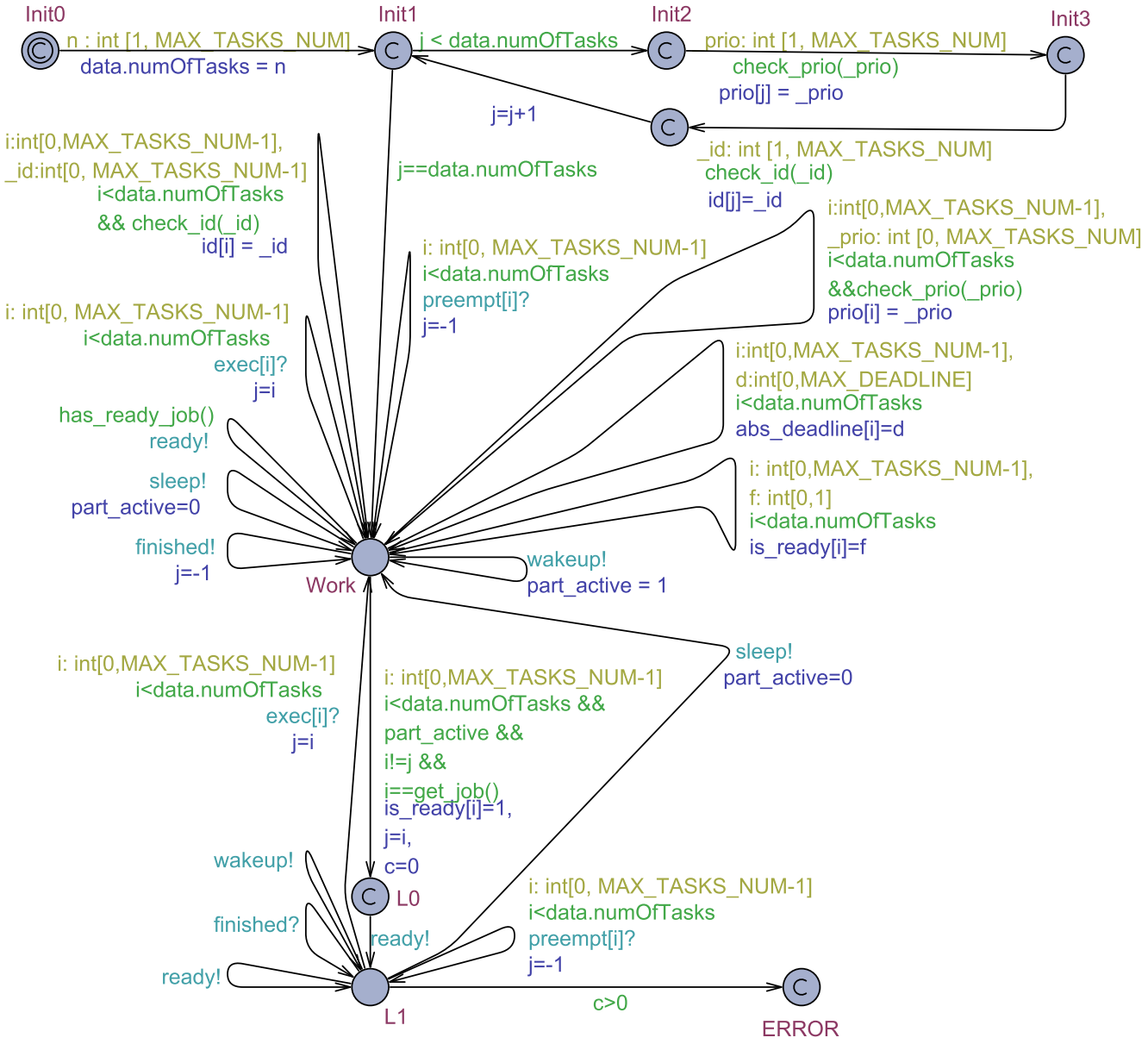

Рисунок В.24 — Автомат-наблюдатель, соответствующий требованию 2 к модели планировщика, работающего по стратегии EDF с вытеснением

В.5.3. Автоматы для требований к модели планировщика с фиксированными приоритетами без вытеснения

Требование 1. *Для любого раздела верно, что если выполнением работ этого раздела управляет планировщик с фиксированными приоритетами без вытеснения, то для постановки на выполнение всегда выбирается работа с максимальным среди всех готовых работ приоритетом*.

Это требование идентично требованию 1 к модели планировщика с фиксированными приоритетами и вытеснением. Следовательно, и автоматы-наблюдатели для проверки этих требований совпадают.

Требование 2. *Для любого раздела верно, что если выполнением работ этого раздела управляет планировщик с фиксированными приоритетами без вытеснения, то любая выполняющаяся*

работа этого раздела может быть вытеснена только при закрытии окна данного раздела. При этом вытеснение происходит мгновенно.

Это требование уточняет требование 5 к моделям планировщиков раздела. Автомат-наблюдатель, соответствующий требованию 2 к модели планировщика с фиксированными приоритетами без вытеснения, приведен на рисунке В.25. Отличие этого автомата-наблюдателя от автомата-наблюдателя для проверки требования 5 к моделям планировщиков раздела состоит лишь в том, что переход из локации $CanNotPreempt$ (вытеснение работы не допускается) в локацию $CanPreempt$ (вытеснение работы допускается) возможен только лишь при выполнении синхронизации $sleep$ (закрытие окна раздела). При выполнении синхронизации по каналу $ready$ текущая локация автомата-наблюдателя остается прежней.

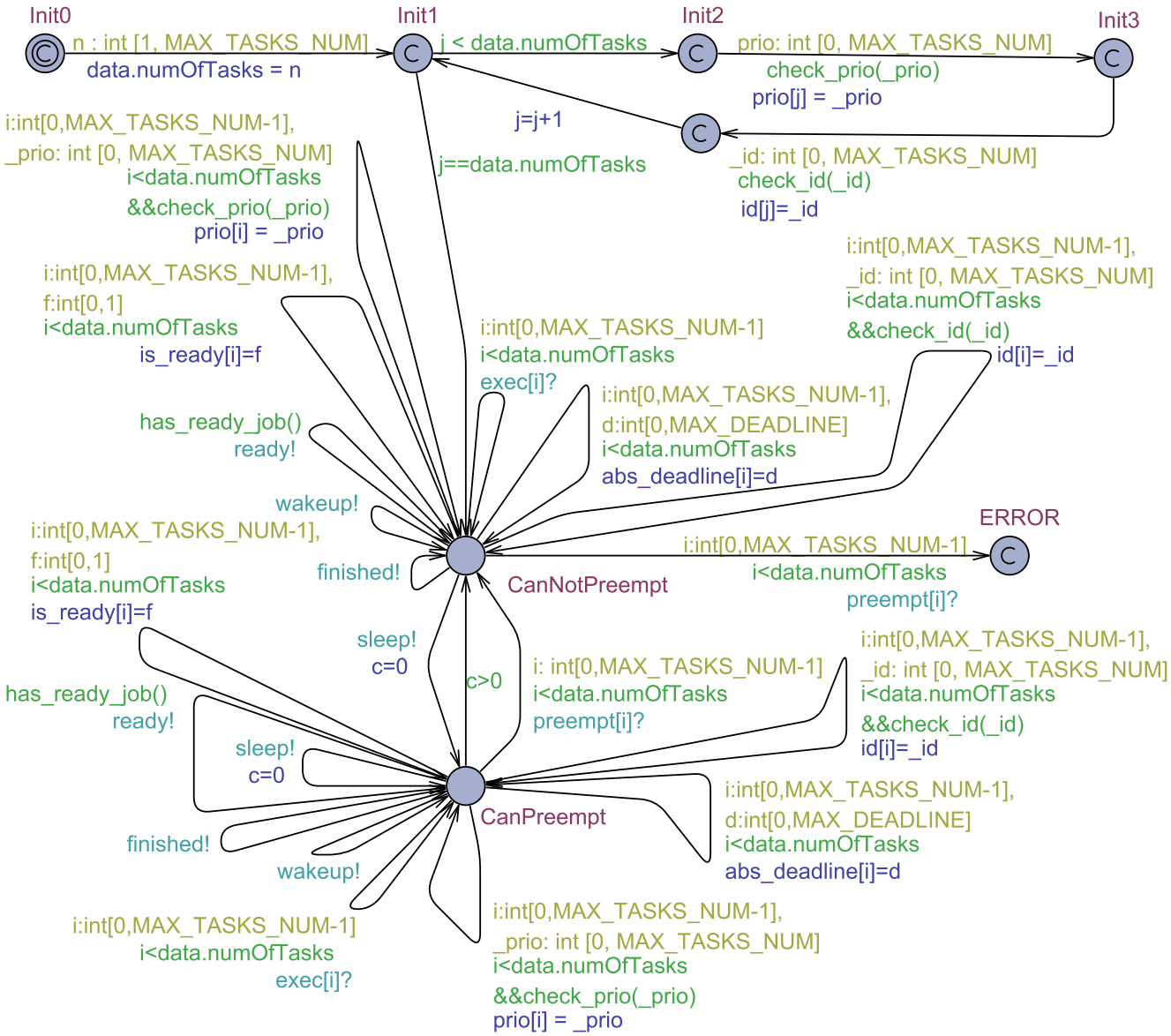

Рисунок В.25 — Автомат-наблюдатель, соответствующий требованию 2 к модели планировщика с фиксированными приоритетами без вытеснения

Требование 3. *Для любого раздела верно, что если выполнением работ этого раздела управляет планировщик, работающий по стратегии EDF без вытеснения, то при появлении новой*

готовой работы некоторая работа мгновенно ставится на выполнение при условии того, что окно этого раздела открыто и на вычислительном ядре нет выполняющихся работ.

Автомат-наблюдатель, соответствующий требованию 3 к модели планировщика с фиксированными приоритетами без вытеснения, приведен на рисунке В.26. Этот автомат-наблюдатель аналогичен автомату-наблюдателю для проверки выполнения требования 2 к модели планировщика с фиксированными приоритетами и вытеснением и отличается лишь условием перехода в локацию $L0$: в автомате-наблюдателе, соответствующем требованию 3 к модели планировщика с фиксированными приоритетами без вытеснения, условием этого перехода является то, что окно раздела открыто ($part\_active$) и отсутствуют выполняющиеся работы ($j == -1$)

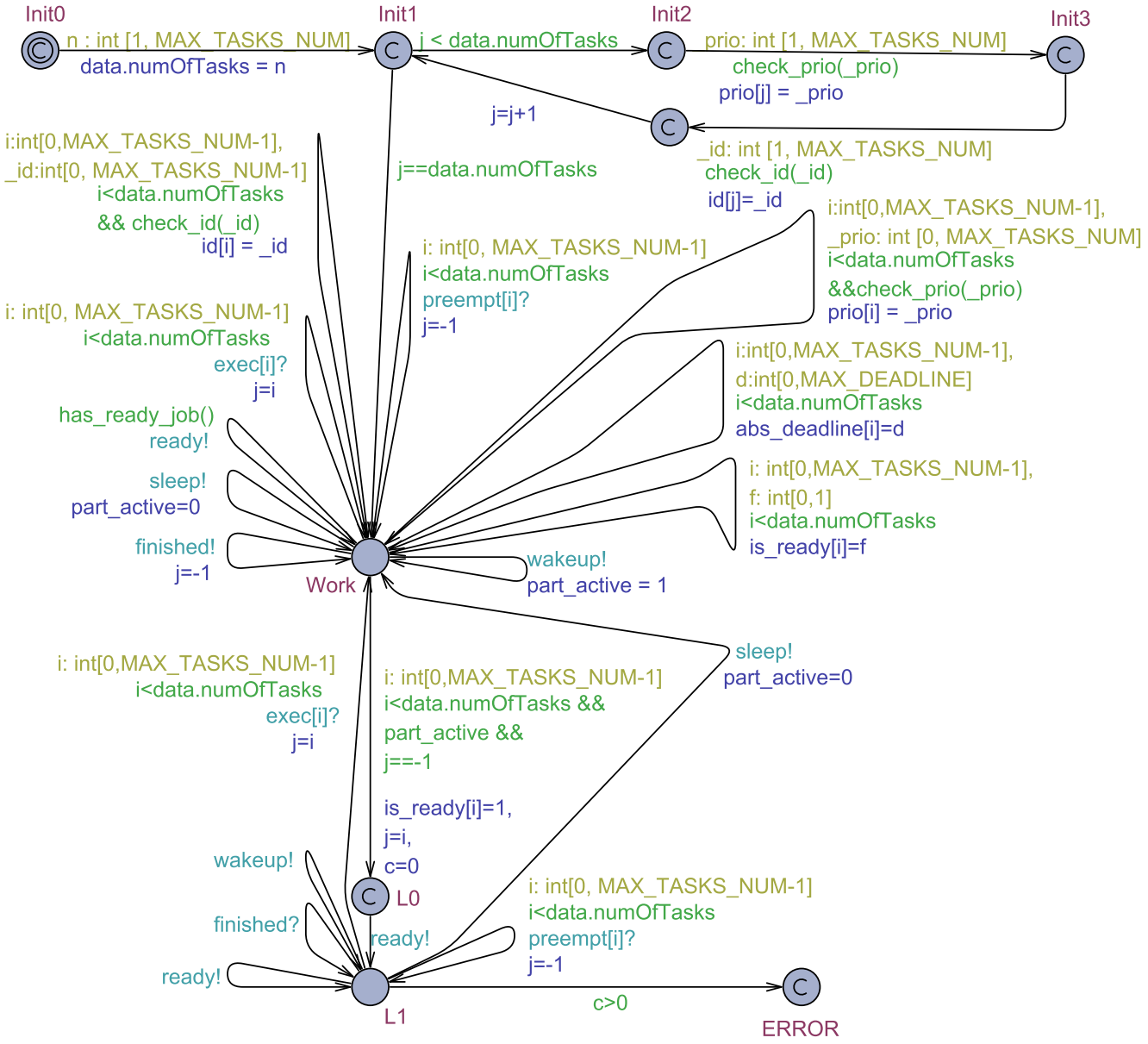

Рисунок В.26 — Автомат-наблюдатель, соответствующий требованию 3 к модели планировщика с фиксированными приоритетами без вытеснения

Приложение Г

Результаты экспериментов

В таблице Г.1 приведены усредненные длительности построения экземпляров модели для конфигураций, принадлежащих различным наборам данных (описания наборов см. в разделе 5.1). Обозначение $x\langle i\rangle$ соответствует i-й конфигурации набора.

Таблица Г.1 — Время построения (в мс) экземпляра модели для конфигураций, принадлежащих различным наборам данных

	x1	x2	x3	x4	x5	x6	x7	x8	x9	x10
Набор 1	7,5	9	10,6	11,8	13,3	14,8	16	17,8	18,8	20,8
Набор 2	7,4	9,2	10,7	12,7	14,2	15,9	17,7	19,8	21,9	23,6
Набор 3	4,8	7,6	10,7	13,9	17,1	20,5	23,9	26,1	31	33,3
Набор 4	27,8	28,7	29,3	29,8	30,1	30,2	30,7	30,7	30,4	30,6
Набор 5	28,9	57,3	82,3	112,4	137,6	163,5	190,1	225,6	250,4	275,9
Набор 6	7,7	14,2	20,9	26,7	32,7	38	43,6	54,1	63,9	69,3
Набор 7	7,5	15	22,6	29,7	37,8	46	53,4	60,8	69,4	75,9
Набор 8	7,2	14,5	22,2	29,2	37,3	44,6	50,8	58,4	67,9	74,8
Набор 9	7,3	14,7	21,9	29,1	37,2	44,9	51,9	58,5	67,2	73,9
Набор 10	7,4	14,6	22,2	29,3	37,3	44,2	50,8	58,2	67,7	75,6

В таблице Г.2 приведены усредненные длительности прогона экземпляров модели для конфигураций, принадлежащих различным наборам данных (описания наборов см. в разделе 5.1).

Таблица Г.2 — Время прогона (в мс) экземпляра модели для конфигураций, принадлежащих различным наборам данных

	x1	x2	x3	x4	x5	x6	x7	x8	x9	x10
Набор 1	1503,8	3907,5	7526,9	13896,3	26254,2	43782,5	65391	88235,3	113673,9	150221,9
Набор 2	1492,8	4985,7	11669,8	26284,7	50592,2	78921,7	109320,6	146174,7	188495,5	243183,7
Набор 3	1378,2	1528,4	1701,6	1845	1987,4	2145,6	2329	2509	2708,6	2845,8
Набор 4	2711,4	3021,4	3183,2	3358,6	3611	3643	3872,8	4085,8	4206,2	4464,8
Набор 5	2716,6	3971	5407,8	6986,6	8663,6	10770,8	12502,6	14302,4	15991,2	18617
Набор 6	1544,2	3089,2	4627,4	6175,4	7702	9216	10767	12407,6	13885,4	15352,8
Набор 7	1495,4	3885,1	5010,7	8698,9	13559,8	13559,8	24037,1	29040,3	34125,5	40404,2
Набор 8	1698,1	4544,9	5584,1	9520,6	15619,2	20583,2	25735,8	31150,5	36848,7	43177,3
Набор 9	1574,4	4065,5	5263,8	8763	14296,6	19042,5	19042,3	29166,9	34529,6	40110,5
Набор 10	1140,7	3234,2	3630,8	6251,7	9965,8	13296,6	16864,6	20508	24403,7	28397,4

В таблице Г.3 приведены усредненные длительности проверки выполнения ограничений реального времени с использованием разработанного средства и с использованием модели, встроенной в САПР «Планировщик ИМА», для конфигураций, принадлежащих 7-му набору данных (описания наборов см. в разделе 5.1).

Таблица Г.3 — Время проверки выполнения ограничений реального времени (в мс) с использованием разработанного средства и с использованием модели, встроенной в САПР «Планировщик ИМА»

	x1	x2	x3	x4	x5	x6	x7	x8	x9	x10
Разработанное средство	1782	4522,2	5816,2	9684,6	15372,2	20271,8	26038	31484,2	38051,8	42767
Модель САПР «Планировщик ИМА»	957	4548,6	11453,2	19594,4	30548,4	42378,6	58426,4	76767	96792,4	128340,8