ACCL Lecture 5: Recursively axiomatizable propositional calculi. Linial – Post Theorem: Undecidability of recognizing axiomatizations of the Classical Propositional Logic

Evgeny Zolin

Department of Mathematical Logic and Theory of Algorithms
Faculty of Mechanics and Mathematics
Moscow State University

Advanced Course in Classical Logic
March 24th, 2021

Axiom system for the CPL

$$
\begin{array}{ll}
(\to 1) & A \to (B \to A) \\
(\to 2) & [A \to (B \to C)] \to [(A \to B) \to (A \to C)] \\
(\wedge 1) & A \wedge B \to A \\
(\wedge 2) & A \wedge B \to B \\
(\wedge 3) & A \to (B \to A \wedge B) \\
(\vee 1) & A \to A \vee B \\
(\vee 2) & B \to A \vee B \\
(\vee 3) & (A \to C) \to [(B \to C) \to (A \vee B \to C)] \\
(\neg 1) & (A \to B) \to [(A \to \neg B) \to \neg A] \\
(\neg 2) & A \to (\neg A \to B) \\
(\neg 3) & \neg\neg A \to A
\end{array}
$$

Axiom system for the CPL

$$
\begin{array}{rl}
(\to 1) & A \to (B \to A) \\
(\to 2) & [A \to (B \to C)] \to [(A \to B) \to (A \to C)] \\
(\wedge 1) & A \wedge B \to A \\
(\wedge 2) & A \wedge B \to B \\
(\wedge 3) & A \to (B \to A \wedge B) \\
(\vee 1) & A \to A \vee B \\
(\vee 2) & B \to A \vee B \\
(\vee 3) & (A \to C) \to [(B \to C) \to (A \vee B \to C)] \\
(\neg 1) & (A \to B) \to [(A \to \neg B) \to \neg A] \\
(\neg 2) & A \to (\neg A \to B) \\
(\neg 3) & \neg\neg A \to A
\end{array}
$$

Rule of inference: modus ponens MP: from A and $A \to B$ we obtain B.

Axiom system for the CPL

$$
\begin{array}{ll}
(\to 1) & A \to (B \to A) \\
(\to 2) & [A \to (B \to C)] \to [(A \to B) \to (A \to C)] \\
(\wedge 1) & A \wedge B \to A \\
(\wedge 2) & A \wedge B \to B \\
(\wedge 3) & A \to (B \to A \wedge B) \\
(\vee 1) & A \to A \vee B \\
(\vee 2) & B \to A \vee B \\
(\vee 3) & (A \to C) \to [(B \to C) \to (A \vee B \to C)] \\
(\neg 1) & (A \to B) \to [(A \to \neg B) \to \neg A] \\
(\neg 2) & A \to (\neg A \to B) \\
(\neg 3) & \neg\neg A \to A
\end{array}
$$

Rule of inference: modus ponens MP: from A and $A \to B$ we obtain B.

Theorem (Completeness)

A formula D is provable from these axioms $\iff$ *D is a tautology.*

Recognizing axiomatizations

Denote by $B_1, \ldots, B_{11}$ the above axioms.

Denote by $B_1, \ldots, B_{11}$ the above axioms.

Let $A_1, \ldots, A_n$ be any formulas.

Recognizing axiomatizations

Denote by $B_1, \ldots, B_{11}$ the above axioms.

Let $A_1, \ldots, A_n$ be any formulas.
Problem. How can we check that the set of formulas $\{A_1, \ldots, A_n\}$ (with the rule MP) axiomatize all tautologies?

Recognizing axiomatizations

Denote by $B_1, \dots, B_{11}$ the above axioms.

Let $A_1, \dots, A_n$ be any formulas.
Problem. How can we check that the set of formulas $\{A_1, \dots, A_n\}$ (with the rule MP) axiomatize all tautologies?

Question. Does there exist an algorithm that solves this problem?

Recognizing axiomatizations

Denote by $B_1, \ldots, B_{11}$ the above axioms.

Let $A_1, \ldots, A_n$ be any formulas.
Problem. How can we check that the set of formulas $\{A_1, \ldots, A_n\}$ (with the rule MP) axiomatize all tautologies?

Question. Does there exist an algorithm that solves this problem?

We need to check two things:

Recognizing axiomatizations

Denote by $B_1, \dots, B_{11}$ the above axioms.

Let $A_1, \dots, A_n$ be any formulas.
Problem. How can we check that the set of formulas $\{A_1, \dots, A_n\}$ (with the rule MP) axiomatize all tautologies?

Question. Does there exist an algorithm that solves this problem?

We need to check two things:

- The set of axioms $\{A_1, \dots, A_n\}$ is correct:
  if D is provable from them, then D is a tautology.

Recognizing axiomatizations

Denote by $B_1, \dots, B_{11}$ the above axioms.

Let $A_1, \dots, A_n$ be any formulas.
Problem. How can we check that the set of formulas $\{A_1, \dots, A_n\}$ (with the rule MP) axiomatize all tautologies?

Question. Does there exist an algorithm that solves this problem?

We need to check two things:

- The set of axioms $\{A_1, \dots, A_n\}$ is correct:
  if D is provable from them, then D is a tautology.
  $\iff$ Each A_i is a tautology.

Recognizing axiomatizations

Denote by $B_1, \ldots, B_{11}$ the above axioms.

Let $A_1, \ldots, A_n$ be any formulas.
Problem. How can we check that the set of formulas $\{A_1, \ldots, A_n\}$ (with the rule MP) axiomatize all tautologies?

Question. Does there exist an algorithm that solves this problem?

We need to check two things:

- The set of axioms $\{A_1, \ldots, A_n\}$ is correct:
  if D is provable from them, then D is a tautology.
  $\Longleftrightarrow$ Each A_i is a tautology. This is a decidable problem.

Recognizing axiomatizations

Denote by $B_1, \ldots, B_{11}$ the above axioms.

Let $A_1, \ldots, A_n$ be any formulas.
Problem. How can we check that the set of formulas $\{A_1, \ldots, A_n\}$ (with the rule MP) axiomatize all tautologies?

Question. Does there exist an algorithm that solves this problem?

We need to check two things:

- The set of axioms $\{A_1, \ldots, A_n\}$ is correct:
  if D is provable from them, then D is a tautology.
  $\Longleftrightarrow$ Each A_i is a tautology. This is a decidable problem.
- The set of axioms $\{A_1, \ldots, A_n\}$ is complete:
  if D is a tautology, then D is provable from them.

Recognizing axiomatizations

Denote by $B_1, \ldots, B_{11}$ the above axioms.

Let $A_1, \ldots, A_n$ be any formulas.
Problem. How can we check that the set of formulas $\{A_1, \ldots, A_n\}$ (with the rule MP) axiomatize all tautologies?

Question. Does there exist an algorithm that solves this problem?

We need to check two things:

- The set of axioms $\{A_1, \ldots, A_n\}$ is correct:
  if D is provable from them, then D is a tautology.
  $\iff$ Each A_i is a tautology. This is a decidable problem.
- The set of axioms $\{A_1, \ldots, A_n\}$ is complete:
  if D is a tautology, then D is provable from them.
  $\iff$ each B_i is provable from $\{A_1, \ldots, A_n\}$.

Recognizing axiomatizations

Denote by $B_1, \dots, B_{11}$ the above axioms.

Let $A_1, \dots, A_n$ be any formulas.
Problem. How can we check that the set of formulas $\{A_1, \dots, A_n\}$ (with the rule MP) axiomatize all tautologies?

Question. Does there exist an algorithm that solves this problem?

We need to check two things:

- The set of axioms $\{A_1, \dots, A_n\}$ is correct:
  if D is provable from them, then D is a tautology.
  $\Longleftrightarrow$ Each A_i is a tautology. This is a decidable problem.
- The set of axioms $\{A_1, \dots, A_n\}$ is complete:
  if D is a tautology, then D is provable from them.
  $\Longleftrightarrow$ each B_i is provable from $\{A_1, \dots, A_n\}$.

In general, an axiomatic system $\{A_1, \dots, A_n\} + (MP)$ may be undecidable.

Recognizing axiomatizations

Denote by $B_1, \ldots, B_{11}$ the above axioms.

Let $A_1, \ldots, A_n$ be any formulas.
Problem. How can we check that the set of formulas $\{A_1, \ldots, A_n\}$ (with the rule MP) axiomatize all tautologies?

Question. Does there exist an algorithm that solves this problem?

We need to check two things:

- The set of axioms $\{A_1, \ldots, A_n\}$ is correct:
  if D is provable from them, then D is a tautology.
  $\iff$ Each A_i is a tautology. This is a decidable problem.
- The set of axioms $\{A_1, \ldots, A_n\}$ is complete:
  if D is a tautology, then D is provable from them.
  $\iff$ each B_i is provable from $\{A_1, \ldots, A_n\}$.

In general, an axiomatic system $\{A_1, \ldots, A_n\} + (MP)$ may be undecidable. Even if it were decidable, there is no guarantee that its algorithm can be built from $\{A_1, \ldots, A_n\}$ effectively.

Other complete axiomatizations

- The set of connectives $\{\to, \neg\}$ is *functionally complete*.
  How to axiomatize all tautologies that use only $\{\to, \neg\}$?

Other complete axiomatizations

- The set of connectives $\{\to, \neg\}$ is *functionally complete*.
  How to axiomatize all tautologies that use only $\{\to, \neg\}$?

Theorem

The axioms $(\to 1)$, $(\to 2)$, $(\neg 1)$, $(\neg 2)$, $(\neg 3)$ *are complete for all tautologies that use only* $\{\to, \neg\}$.

Other complete axiomatizations

- The set of connectives $\{\to, \neg\}$ is *functionally complete*.
  How to axiomatize all tautologies that use only $\{\to, \neg\}$?

Theorem

The axioms $(\to 1)$, $(\to 2)$, $(\neg 1)$, $(\neg 2)$, $(\neg 3)$ *are complete for all tautologies that use only* $\{\to, \neg\}$.

- What about $\{\to\}$? It is functionally incomplete. But
  How to axiomatize all tautologies that use only $\to$?

Other complete axiomatizations

- The set of connectives $\{\to, \neg\}$ is *functionally complete*.
  How to axiomatize all tautologies that use only $\{\to, \neg\}$?

Theorem

The axioms $(\to 1)$, $(\to 2)$, $(\neg 1)$, $(\neg 2)$, $(\neg 3)$ *are complete for all tautologies that use only* $\{\to, \neg\}$.

- What about $\{\to\}$? It is functionally incomplete. But
  How to axiomatize all tautologies that use only $\to$?
  (!) Axioms $(\to 1)$ and $(\to 2)$ are not enough! We need:

  Pierce's law: $((A \to B) \to A) \to A$

Other complete axiomatizations

- The set of connectives $\{\to, \neg\}$ is *functionally complete*.
  How to axiomatize all tautologies that use only $\{\to, \neg\}$?

Theorem

The axioms $(\to 1)$, $(\to 2)$, $(\neg 1)$, $(\neg 2)$, $(\neg 3)$ *are complete for all tautologies that use only* $\{\to, \neg\}$.

- What about $\{\to\}$? It is functionally incomplete. But
  How to axiomatize all tautologies that use only $\to$?
  (!) Axioms $(\to 1)$ and $(\to 2)$ are not enough! We need:

  Pierce's law: $((A \to B) \to A) \to A$

- Can we axiomatize all $\to$-tautologies with only one axiom?

Other complete axiomatizations

- The set of connectives $\{\to, \neg\}$ is *functionally complete*.
  How to axiomatize all tautologies that use only $\{\to, \neg\}$?

Theorem

The axioms $(\to 1)$, $(\to 2)$, $(\neg 1)$, $(\neg 2)$, $(\neg 3)$ *are complete for all tautologies that use only* $\{\to, \neg\}$.

- What about $\{\to\}$? It is functionally incomplete. But
  How to axiomatize all tautologies that use only $\to$?
  (!) Axioms $(\to 1)$ and $(\to 2)$ are not enough! We need:

  Pierce's law: $((A \to B) \to A) \to A$

- Can we axiomatize all $\to$-tautologies with only one axiom? Yes:
  $[(A \to B) \to C] \to [(C \to A) \to (D \to A)]$. (Łukasiewicz, 1948).

Other complete axiomatizations

- The set of connectives $\{\to, \neg\}$ is *functionally complete*.
  How to axiomatize all tautologies that use only $\{\to, \neg\}$?

Theorem

The axioms $(\to 1)$, $(\to 2)$, $(\neg 1)$, $(\neg 2)$, $(\neg 3)$ *are complete for all tautologies that use only* $\{\to, \neg\}$.

- What about $\{\to\}$? It is functionally incomplete. But
  How to axiomatize all tautologies that use only $\to$?
  (!) Axioms $(\to 1)$ and $(\to 2)$ are not enough! We need:

  Pierce's law: $((A \to B) \to A) \to A$

- Can we axiomatize all $\to$-tautologies with only one axiom? Yes:

  $[(A \to B) \to C] \to [(C \to A) \to (D \to A)]$. (Łukasiewicz, 1948).

- One axiom can be built for $\{\to, \neg\}$, for $\{\neg, \wedge, \vee, \to\}$, for $\{\leftrightarrow\}$, etc.

Other complete axiomatizations

- The set of connectives $\{\to, \neg\}$ is *functionally complete*.
  How to axiomatize all tautologies that use only $\{\to, \neg\}$?

Theorem

The axioms $(\to 1)$, $(\to 2)$, $(\neg 1)$, $(\neg 2)$, $(\neg 3)$ *are complete for all tautologies that use only* $\{\to, \neg\}$.

- What about $\{\to\}$? It is functionally incomplete. But
  How to axiomatize all tautologies that use only $\to$?
  (!) Axioms $(\to 1)$ and $(\to 2)$ are not enough! We need:

  Pierce's law: $((A \to B) \to A) \to A$

- Can we axiomatize all $\to$-tautologies with only one axiom? Yes:

  $[(A \to B) \to C] \to [(C \to A) \to (D \to A)]$. (Łukasiewicz, 1948).

- One axiom can be built for $\{\to, \neg\}$, for $\{\neg, \wedge, \vee, \to\}$, for $\{\leftrightarrow\}$, etc.
- **Ted Ulrich** — collects single axioms for many logics in $\{\to\}$ and $\{\leftrightarrow\}$.
  `https://web.ics.purdue.edu/~dulrich/Home-page.htm`

Decidable and semi-decidable sets

Let U be $\mathbb{N}$ or Σ^* (the set of all words over an alphabet Σ).

Definition 1. A set $D \subseteq U$ is called decidable (or recursive) if its characteristic function is computable:

$$\forall x \in \Sigma^* \qquad \chi_D(x) \;=\; \begin{cases} 1, & x \in D, \\ 0, & x \notin D. \end{cases}$$

Decidable and semi-decidable sets

Let U be $\mathbb{N}$ or Σ^* (the set of all words over an alphabet Σ).

Definition 1. A set $D \subseteq U$ is called decidable (or recursive) if its characteristic function is computable:

$$\forall x \in \Sigma^* \qquad \chi_D(x) \;=\; \begin{cases} 1, & x \in D, \\ 0, & x \notin D. \end{cases}$$

Definition 2. A set $D \subseteq U$ is called semi-decidable (or recursively enumerable, r.e.) if its semi-characteristic function is computable:

$$\forall x \in \Sigma^* \qquad \chi_D(x) \;=\; \begin{cases} 1, & x \in D, \\ \mathit{undefined}, & x \notin D. \end{cases}$$

Decidable and semi-decidable sets

Let U be $\mathbb{N}$ or Σ^* (the set of all words over an alphabet Σ).

Definition 1. A set $D \subseteq U$ is called decidable (or recursive) if its characteristic function is computable:

$$\forall x \in \Sigma^* \qquad \chi_D(x) = \begin{cases} 1, & x \in D, \\ 0, & x \notin D. \end{cases}$$

Definition 2. A set $D \subseteq U$ is called semi-decidable (or recursively enumerable, r.e.) if its semi-characteristic function is computable:

$$\forall x \in \Sigma^* \qquad \chi_D(x) = \begin{cases} 1, & x \in D, \\ \mathit{undefined}, & x \notin D. \end{cases}$$

Example

Prime numbers — decidable.

Decidable and semi-decidable sets

Let U be $\mathbb{N}$ or Σ^* (the set of all words over an alphabet Σ).

Definition 1. A set $D \subseteq U$ is called decidable (or recursive) if its characteristic function is computable:

$$\forall x \in \Sigma^* \qquad \chi_D(x) \;=\; \begin{cases} 1, & x \in D, \\ 0, & x \notin D. \end{cases}$$

Definition 2. A set $D \subseteq U$ is called semi-decidable (or recursively enumerable, r.e.) if its semi-characteristic function is computable:

$$\forall x \in \Sigma^* \qquad \chi_D(x) \;=\; \begin{cases} 1, & x \in D, \\ \mathit{undefined}, & x \notin D. \end{cases}$$

Example

Prime numbers — decidable. Tautologies — decidable.

Decidable and semi-decidable sets

Let U be $\mathbb{N}$ or Σ^* (the set of all words over an alphabet Σ).

Definition 1. A set $D \subseteq U$ is called decidable (or recursive) if its characteristic function is computable:

$$\forall x \in \Sigma^* \qquad \chi_D(x) = \begin{cases} 1, & x \in D, \\ 0, & x \notin D. \end{cases}$$

Definition 2. A set $D \subseteq U$ is called semi-decidable (or recursively enumerable, r.e.) if its semi-characteristic function is computable:

$$\forall x \in \Sigma^* \qquad \chi_D(x) = \begin{cases} 1, & x \in D, \\ \mathit{undefined}, & x \notin D. \end{cases}$$

Example

Prime numbers — decidable. Tautologies — decidable.
$\{n \in \mathbb{N} \mid \exists a, b, c \in \mathbb{N}\colon\ n = a^4 + b^4 - c^4\}$ — semi-decidable.

Decidable and semi-decidable sets

Let U be $\mathbb{N}$ or Σ^* (the set of all words over an alphabet Σ).

Definition 1. A set $D \subseteq U$ is called decidable (or recursive) if its characteristic function is computable:

$$\forall x \in \Sigma^* \qquad \chi_D(x) \;=\; \begin{cases} 1, & x \in D, \\ 0, & x \notin D. \end{cases}$$

Definition 2. A set $D \subseteq U$ is called semi-decidable (or recursively enumerable, r.e.) if its semi-characteristic function is computable:

$$\forall x \in \Sigma^* \qquad \chi_D(x) \;=\; \begin{cases} 1, & x \in D, \\ \mathit{undefined}, & x \notin D. \end{cases}$$

Example

Prime numbers — decidable. Tautologies — decidable.
$\{n \in \mathbb{N} \mid \exists a, b, c \in \mathbb{N}\colon\ n = a^4 + b^4 - c^4\}$ — semi-decidable. Decidable?

A calculus consists of: 1) a set of axioms, 2) a set of rules.

Recursively enumerable calculi

A calculus consists of: 1) a set of axioms, 2) a set of rules.
A rule with n premises — a relation $R \subseteq \mathrm{Fm}^n \times \mathrm{Fm}$.
(usually a partial function, usually a computable function)

Recursively enumerable calculi

A calculus consists of: 1) a set of axioms, 2) a set of rules.
A rule with n premises — a relation $R \subseteq \mathrm{Fm}^n \times \mathrm{Fm}$.
(usually a partial function, usually a computable function)
A calculus is called decidable / semi-decidable if the set of its theorems (provable formulas) is decidable / semi-decidable.

Recursively enumerable calculi

A calculus consists of: 1) a set of axioms, 2) a set of rules.
A rule with n premises — a relation $R \subseteq \mathrm{Fm}^n \times \mathrm{Fm}$.
(usually a partial function, usually a computable function)
A calculus is called decidable / semi-decidable if the set of its theorems (provable formulas) is decidable / semi-decidable.

Theorem 1

Any calculus with a finite set of axioms and a finite set of rules is semi-decidable.

Recursively enumerable calculi

A calculus consists of: 1) a set of axioms, 2) a set of rules.
A rule with n premises — a relation $R \subseteq \mathrm{Fm}^n \times \mathrm{Fm}$.
(usually a partial function, usually a computable function)
A calculus is called decidable / semi-decidable if the set of its theorems (provable formulas) is decidable / semi-decidable.

Theorem 1

Any calculus with a finite set of axioms and a finite set of rules is semi-decidable.

Proof.

Given a formula D, build all possible proofs and search for D. □

Recursively enumerable calculi

Theorem 1

Any calculus with a finite set of axioms and a finite set of rules is semi-decidable.

Proof.

Given a formula D, build all possible proofs and search for D. $\square$

Recursively enumerable calculi

Theorem 1

Any calculus with a finite set of axioms and a finite set of rules is semi-decidable.

Proof.

Given a formula D, build all possible proofs and search for D. $\square$

Theorem 2

Any calculus with a decidable set of axioms and a finite set of rules is semi-decidable.

Recursively enumerable calculi

Theorem 1

Any calculus with a finite set of axioms and a finite set of rules is semi-decidable.

Proof.

Given a formula D, build all possible proofs and search for D. $\square$

Theorem 2

Any calculus with a decidable set of axioms and a finite set of rules is semi-decidable.

Proof — the same.

Recursively enumerable calculi

Theorem 1

Any calculus with a finite set of axioms and a finite set of rules is semi-decidable.

Proof.

Given a formula D, build all possible proofs and search for D. $\square$

Theorem 2

Any calculus with a decidable set of axioms and a finite set of rules is semi-decidable.

Proof — the same.

Theorem 3

Any calculus with a semi-decidable set of axioms and a finite set of rules is semi-decidable.

Recursively enumerable calculi

Theorem 3

Any calculus with a semi-decidable set of axioms and a finite set of rules is semi-decidable.

Recursively enumerable calculi

Theorem 3

Any calculus with a semi-decidable set of axioms and a finite set of rules is semi-decidable.

Proof.

There is a computable enumeration $A_0, A_1, \ldots$ of all axioms of this calculus.

Recursively enumerable calculi

Theorem 3

Any calculus with a semi-decidable set of axioms and a finite set of rules is semi-decidable.

Proof.

There is a computable enumeration $A_0, A_1, \ldots$ of all axioms of this calculus.
Given a formula D, is it provable?

Recursively enumerable calculi

Theorem 3

Any calculus with a semi-decidable set of axioms and a finite set of rules is semi-decidable.

Proof.

There is a computable enumeration $A_0, A_1, \ldots$ of all axioms of this calculus.
Given a formula D, is it provable?
Algorithm: for each $n \in \mathbb{N}$,
– build the formulas $A_0, \ldots, A_n$,

Recursively enumerable calculi

Theorem 3

Any calculus with a semi-decidable set of axioms and a finite set of rules is semi-decidable.

Proof.

There is a computable enumeration $A_0, A_1, \ldots$ of all axioms of this calculus.
Given a formula D, is it provable?
Algorithm: for each $n \in \mathbb{N}$,
– build the formulas $A_0, \ldots, A_n$,
– build proofs of length $\leqslant n$ from $A_0, \ldots, A_n$,

Recursively enumerable calculi

Theorem 3

Any calculus with a semi-decidable set of axioms and a finite set of rules is semi-decidable.

Proof.

There is a computable enumeration $A_0, A_1, \ldots$ of all axioms of this calculus.
Given a formula D, is it provable?
Algorithm: for each $n \in \mathbb{N}$,
– build the formulas $A_0, \ldots, A_n$,
– build proofs of length $\leqslant n$ from $A_0, \ldots, A_n$,
– check if you obtained D. If not, continue for $n + 1$. □

Recursively enumerable calculi

Theorem 3

Any calculus with a semi-decidable set of axioms and a finite set of rules is semi-decidable.

Proof.

There is a computable enumeration $A_0, A_1, \ldots$ of all axioms of this calculus.
Given a formula D, is it provable?
Algorithm: for each $n \in \mathbb{N}$,
– build the formulas $A_0, \ldots, A_n$,
– build proofs of length $\leqslant n$ from $A_0, \ldots, A_n$,
– check if you obtained D. If not, continue for $n+1$. $\square$

Question: Theorem 2 (decidable set of axioms) and Theorem 3 (semi-decidable set of axioms) talk about the same calculi?

The problem of checking axiomatizations

Assume that we have a finite set of tautologies $A_1, \ldots, A_n$.

The problem of checking axiomatizations

Assume that we have a finite set of tautologies $A_1, \ldots, A_n$.

The rules of inference are always

$$\text{MP } \frac{A \quad A \to B}{B} \qquad \text{and} \qquad \text{Substitution rule } \frac{A(p_1, \ldots, p_k)}{A(B_1, \ldots, B_k)}$$

The problem of checking axiomatizations

Assume that we have a finite set of tautologies $A_1, \ldots, A_n$.

The rules of inference are always

$$\text{MP } \frac{A \quad A \to B}{B} \qquad \text{and} \qquad \text{Substitution rule } \frac{A(p_1, \ldots, p_k)}{A(B_1, \ldots, B_k)}$$

Problem

Does there exist an **algorithm** *that checks whether*

the given tautologies $A_1, \ldots, A_n$ axiomatize all tautologies?

The problem of checking axiomatizations

Assume that we have a finite set of tautologies $A_1, \ldots, A_n$.

The rules of inference are always

$$\text{MP } \frac{A \quad A \to B}{B} \qquad \text{and} \qquad \text{Substitution rule } \frac{A(p_1, \ldots, p_k)}{A(B_1, \ldots, B_k)}$$

Problem

Does there exist an **algorithm** *that checks whether*

the given tautologies $A_1, \ldots, A_n$ axiomatize all tautologies?

Theorem (Samuel Linial, Emil Post, 1949)

There is **no** *such an algorithm.*

How can we prove undecidability? Reduction!

Problem (or language) — any language $L \subseteq \Sigma^*$.

How can we prove undecidability? Reduction!

Problem (or language) — any language $L \subseteq \Sigma^*$.

Definition (Algorithmic (effective) reduction, $L_1 \preccurlyeq L_2$)

A problem $L_1 \subseteq \Sigma_1^*$ is reducible to a problem $L_2 \subseteq \Sigma_2^*$ if

How can we prove undecidability? Reduction!

Problem (or language) — any language $L \subseteq \Sigma^*$.

Definition (Algorithmic (effective) reduction, $L_1 \preccurlyeq L_2$)

A problem $L_1 \subseteq \Sigma_1^*$ is reducible to a problem $L_2 \subseteq \Sigma_2^*$ if there is a computable function $f\colon \Sigma_1^* \to \Sigma_2^*$ such that:

How can we prove undecidability? Reduction!

Problem (or language) — any language $L \subseteq \Sigma^*$.

Definition (Algorithmic (effective) reduction, $L_1 \preccurlyeq L_2$)

A problem $L_1 \subseteq \Sigma_1^*$ is reducible to a problem $L_2 \subseteq \Sigma_2^*$ if there is a computable function $f\colon \Sigma_1^* \to \Sigma_2^*$ such that:

$$\text{for all words } x \in \Sigma_1^*: \qquad x \in L_1 \iff f(x) \in L_2.$$

How can we prove undecidability? Reduction!

Problem (or language) — any language $L \subseteq \Sigma^*$.

Definition (Algorithmic (effective) reduction, $L_1 \preccurlyeq L_2$)

A problem $L_1 \subseteq \Sigma_1^*$ is reducible to a problem $L_2 \subseteq \Sigma_2^*$ if there is a computable function $f\colon \Sigma_1^* \to \Sigma_2^*$ such that:

$$\text{for all words } x \in \Sigma_1^*: \qquad x \in L_1 \iff f(x) \in L_2.$$

Lemma

(a) *$L \preccurlyeq L'$ and L' is decidable* $\implies$ *L is decidable;*
(b) *$L \preccurlyeq L'$ and L is undecidable* $\implies$ *L' is undecidable.*

How can we prove undecidability? Reduction!

Problem (or language) — any language $L \subseteq \Sigma^*$.

Definition (Algorithmic (effective) reduction, $L_1 \preccurlyeq L_2$)

A problem $L_1 \subseteq \Sigma_1^*$ is reducible to a problem $L_2 \subseteq \Sigma_2^*$ if there is a computable function $f\colon \Sigma_1^* \to \Sigma_2^*$ such that:

$$\text{for all words } x \in \Sigma_1^*: \qquad x \in L_1 \iff f(x) \in L_2.$$

Lemma

(a) $L \preccurlyeq L'$ *and* L' *is decidable* $\implies$ L *is decidable;*
(b) $L \preccurlyeq L'$ *and* L *is undecidable* $\implies$ L' *is undecidable.*

Proof.

If the machine $M'(y)$ decides the problem $y \in L'$,
then the machine $M(x) = M'(f(x))$ decides the problem $x \in L$. $\square$

Uniform Post Production System (Tag system)

Tag system — $\Pi = (\Sigma, P, \ell)$ consists of:

– $\Sigma = (a_1, \ldots, a_m)$ — a finite alphabet,

Uniform Post Production System (Tag system)

Tag system — $\Pi = (\Sigma, P, \ell)$ consists of:

– $\Sigma = (a_1, \ldots, a_m)$ — a finite alphabet,

– $P = (x_1, \ldots, x_m)$ — m words,

Uniform Post Production System (Tag system)

Tag system — $\Pi = (\Sigma, P, \ell)$ consists of:

– $\Sigma = (a_1, \ldots, a_m)$ — a finite alphabet,

– $P = (x_1, \ldots, x_m)$ — m words, $\ell \geqslant 1$ — an integer.

Uniform Post Production System (Tag system)

Tag system — $\Pi = (\Sigma, P, \ell)$ consists of:
– $\Sigma = (a_1, \ldots, a_m)$ — a finite alphabet,
– $P = (x_1, \ldots, x_m)$ — m words, $\ell \geqslant 1$ — an integer.

Π is applicable to a word w if $|w| \geqslant \ell$.

Uniform Post Production System (Tag system)

Tag system — $\Pi = (\Sigma, P, \ell)$ consists of:

– $\Sigma = (a_1, \ldots, a_m)$ — a finite alphabet,

– $P = (x_1, \ldots, x_m)$ — m words, $\ell \geqslant 1$ — an integer.

Π is applicable to a word w if $|w| \geqslant \ell$.

Application of Π to w: $\boxed{a_i\, z\, v \xrightarrow{\Pi} v\, x_i}$ where $|a_i z| = \ell$.

Uniform Post Production System (Tag system)

Tag system — $\Pi = (\Sigma, P, \ell)$ consists of:
– $\Sigma = (a_1, \ldots, a_m)$ — a finite alphabet,
– $P = (x_1, \ldots, x_m)$ — m words, $\ell \geqslant 1$ — an integer.

Π is applicable to a word w if $|w| \geqslant \ell$.

Application of Π to w: $\boxed{a_i\, z\, v \xrightarrow{\Pi} v\, x_i}$ where $|a_i z| = \ell$. So,
– remember the first symbol a_i of w;

Uniform Post Production System (Tag system)

Tag system — $\Pi = (\Sigma, P, \ell)$ consists of:
– $\Sigma = (a_1, \ldots, a_m)$ — a finite alphabet,
– $P = (x_1, \ldots, x_m)$ — m words, $\ell \geqslant 1$ — an integer.

Π is applicable to a word w if $|w| \geqslant \ell$.

Application of Π to w: $\boxed{a_i\, z\, v \xrightarrow{\Pi} v\, x_i}$ where $|a_i z| = \ell$. So,
– remember the first symbol a_i of w;
– delete the first ℓ symbols from w;

Uniform Post Production System (Tag system)

Tag system — $\Pi = (\Sigma, P, \ell)$ consists of:

– $\Sigma = (a_1, \ldots, a_m)$ — a finite alphabet,

– $P = (x_1, \ldots, x_m)$ — m words, $\ell \geqslant 1$ — an integer.

Π is applicable to a word w if $|w| \geqslant \ell$.

Application of Π to w: $\boxed{a_i\, z\, v \xrightarrow{\Pi} v\, x_i}$ where $|a_i z| = \ell$. So,

– remember the first symbol a_i of w;

– delete the first ℓ symbols from w;

– add the word x_i to the end.

Uniform Post Production System (Tag system)

Tag system — $\Pi = (\Sigma, P, \ell)$ consists of:
– $\Sigma = (a_1, \dots, a_m)$ — a finite alphabet,
– $P = (x_1, \dots, x_m)$ — m words, $\ell \geqslant 1$ — an integer.

Π is applicable to a word w if $|w| \geqslant \ell$.

Application of Π to w: $\boxed{a_i\, z\, v \xrightarrow{\Pi} v\, x_i}$ where $|a_i z| = \ell$. So,
– remember the first symbol a_i of w;
– delete the first ℓ symbols from w;
– add the word x_i to the end.

Definition

Π halts on an input word w if $w \xrightarrow{\Pi} w_0 \xrightarrow{\Pi} \dots u$ for some word $|u| < \ell$.

Uniform Post Production System (Tag system)

Tag system — $\Pi = (\Sigma, P, \ell)$ consists of:
– $\Sigma = (a_1, \ldots, a_m)$ — a finite alphabet,
– $P = (x_1, \ldots, x_m)$ — m words, $\ell \geqslant 1$ — an integer.

Π is applicable to a word w if $|w| \geqslant \ell$.

Application of Π to w: $\boxed{a_i\, z\, v \xrightarrow{\Pi} v\, x_i}$ where $|a_i z| = \ell$. So,
– remember the first symbol a_i of w;
– delete the first ℓ symbols from w;
– add the word x_i to the end.

Definition

Π halts on an input word w if $w \xrightarrow{\Pi} w_0 \xrightarrow{\Pi} \ldots u$ for some word $|u| < \ell$.

Theorem (Minski, 1961)

The problem "Π stops on w" is **undecidable**.

Uniform Post Production System (Tag system)

Tag system — $\Pi = (\Sigma, P, \ell)$ consists of:
– $\Sigma = (a_1, \ldots, a_m)$ — a finite alphabet,
– $P = (x_1, \ldots, x_m)$ — m words, $\ell \geqslant 1$ — an integer.

Π is applicable to a word w if $|w| \geqslant \ell$.

Application of Π to w: $\boxed{a_i\, z\, v \xrightarrow{\Pi} v\, x_i}$ where $|a_i z| = \ell$. So,
– remember the first symbol a_i of w;
– delete the first ℓ symbols from w;
– add the word x_i to the end.

Definition

Π halts on an input word w if $w \xrightarrow{\Pi} w_0 \xrightarrow{\Pi} \ldots u$ for some word $|u| < \ell$.

Theorem (Minski, 1961)

The problem "Π stops on w" is **undecidable**.

Moreover, $\exists \Pi_0$ such that the problem "Π_0 stops on w" is undecidable.

Undecidability of axiomatizations of CPL

Theorem (Samuel Linial, Emil Post, 1949)

There is **no** *algorithm that checks:*

whether the given tautologies $A_1, \ldots, A_n$ axiomatize all tautologies?

Undecidability of axiomatizations of CPL

Theorem (Samuel Linial, Emil Post, 1949)

There is **no** *algorithm that checks:*

whether the given tautologies $A_1, \ldots, A_n$ axiomatize all tautologies?

Proof (sketch).

Undecidability of axiomatizations of CPL

Theorem (Samuel Linial, Emil Post, 1949)

There is **no** *algorithm that checks:*

whether the given tautologies $A_1, \ldots, A_n$ axiomatize all tautologies?

Proof (sketch).

To a symbol a_3, we build a formula $E_3 = p \to (p \to (p \to (p \to p)))$.

Undecidability of axiomatizations of CPL

Theorem (Samuel Linial, Emil Post, 1949)

There is **no** *algorithm that checks:*

whether the given tautologies $A_1, \ldots, A_n$ axiomatize all tautologies?

Proof (sketch).

To a symbol a_3, we build a formula $E_3 = p \to (p \to (p \to (p \to p)))$.

We encode a word $a_2 a_5 a_3 a_1$ into a formula $E_2 \wedge (E_5 \wedge (E_3 \wedge E_1)))$.

Theorem (Samuel Linial, Emil Post, 1949)

There is **no** *algorithm that checks:*

whether the given tautologies $A_1, \ldots, A_n$ axiomatize all tautologies?

Proof (sketch).

To a symbol a_3, we build a formula $E_3 = p \to (p \to (p \to (p \to p)))$.

We encode a word $a_2 a_5 a_3 a_1$ into a formula $E_2 \wedge (E_5 \wedge (E_3 \wedge E_1)))$.

Given a Post system Π of and an input word w,

Undecidability of axiomatizations of CPL

Theorem (Samuel Linial, Emil Post, 1949)

There is **no** *algorithm that checks:*

whether the given tautologies $A_1, \ldots, A_n$ axiomatize all tautologies?

Proof (sketch).

To a symbol a_3, we build a formula $E_3 = p \to (p \to (p \to (p \to p)))$.

We encode a word $a_2 a_5 a_3 a_1$ into a formula $E_2 \wedge (E_5 \wedge (E_3 \wedge E_1)))$.

Given a Post system Π of and an input word w,

an algorithm builds a calculus $C = C(\Pi, w) = \{A_1, \ldots, A_n\}$ such that

Undecidability of axiomatizations of CPL

Theorem (Samuel Linial, Emil Post, 1949)

There is **no** *algorithm that checks:*

whether the given tautologies $A_1, \ldots, A_n$ axiomatize all tautologies?

Proof (sketch).

To a symbol a_3, we build a formula $E_3 = p \to (p \to (p \to (p \to p)))$.

We encode a word $a_2 a_5 a_3 a_1$ into a formula $E_2 \wedge (E_5 \wedge (E_3 \wedge E_1)))$.

Given a Post system Π of and an input word w,

an algorithm builds a calculus $C = C(\Pi, w) = \{A_1, \ldots, A_n\}$ such that

Π stops on input $w \iff$ the calculus C can prove all tautologies.

Undecidability of axiomatizations of CPL

Theorem (Samuel Linial, Emil Post, 1949)

There is **no** *algorithm that checks:*

whether the given tautologies $A_1, \ldots, A_n$ axiomatize all tautologies?

Proof (sketch).

To a symbol a_3, we build a formula $E_3 = p \to (p \to (p \to (p \to p)))$.

We encode a word $a_2 a_5 a_3 a_1$ into a formula $E_2 \wedge (E_5 \wedge (E_3 \wedge E_1)))$.

Given a Post system Π of and an input word w,

an algorithm builds a calculus $C = C(\Pi, w) = \{A_1, \ldots, A_n\}$ such that

Π stops on input w $\iff$ the calculus C can prove all tautologies.

This is a reduction of the halting problem for Tag systems to our problem.

$\square$

New results (Grigory Bokov, 2016, MSU)

Theorem

Fix any calculus $D_0 = \{B_1, \ldots, B_m\}$ *(even in* $\to$*!).*

Then the following problem is **undecidable***:*

$$\text{given a calculus } C = \{A_1, \ldots, A_n\}, \text{ decide if } C \supseteq D.$$

New results (Grigory Bokov, 2016, MSU)

Theorem

Fix any calculus $D_0 = \{B_1, \ldots, B_m\}$ (even in $\to$!).
Then the following problem is **undecidable***:*

$$\text{given a calculus } C = \{A_1, \ldots, A_n\}, \text{ decide if } C \supseteq D.$$

Corollary

Fix any formula B. Then the following problem is **undecidable***:*

$$\text{given a calculus } C = \{A_1, \ldots, A_n\}, \text{ decide if } C \vdash B.$$

New results (Grigory Bokov, 2016, MSU)

Theorem

Fix any calculus $D_0 = \{B_1, \dots, B_m\}$ (even in $\to$!).
Then the following problem is **undecidable**:

$$\text{given a calculus } C = \{A_1, \dots, A_n\}, \text{ decide if } C \supseteq D.$$

Corollary

Fix any formula B. Then the following problem is **undecidable**:

$$\text{given a calculus } C = \{A_1, \dots, A_n\}, \text{ decide if } C \vdash B.$$

Theorem

Fix a calculus $D = \{B_1, \dots, B_m\}$ (even in $\to$!) with $D \vdash p \to (q \to p)$.
Then the following problem is **undecidable**:

$$\text{given a calculus } C = \{A_1, \dots, A_n\}, \text{ decide if } C = D.$$