Turing Degrees and the Friedberg-Muchnik Theorem

Mathematical Logic for Computer Science

Master’s Degree in Computer Science

Simone Bianco (1986936)

Academic Year 2024/2025

SAPIENZA
UNIVERSITÀ DI ROMA

Table of Contents

- ▶ Introduction
- ▶ Degrees of Unsolvability
- ▶ Post's problem

Introduction

Notation

$\Phi^A_{i,s}(x)$ denotes the **computation** of the TM M_i for $s \in \mathbb{N}$ steps on input $x \in \mathbb{N}$ while having access to an oracle for the subset $A \subseteq \mathbb{N}$

- $\Phi^A_{i,s}(x) \downarrow$: the computation halts after s steps
- $\Phi^A_i(x) \downarrow$: there is a $s \in \mathbb{N}$ such that $\Phi^A_{i,s}(x) \downarrow$
- $\Phi^A_i(x) \uparrow$: there is no $s \in \mathbb{N}$ such that $\Phi^A_{i,s}(x) \downarrow$
- $\phi^A_i(x)$: output of the computation (defined iff $\Phi^A_i(x) \downarrow$)

Introduction
Notation

$\Phi^A_{i,s}(x)$ denotes the **computation** of the TM M_i for $s \in \mathbb{N}$ steps on input $x \in \mathbb{N}$ while having access to an oracle for the subset $A \subseteq \mathbb{N}$

- $\Phi^A_{i,s}(x) \downarrow$: the computation halts after s steps
- $\Phi^A_i(x) \downarrow$: there is a $s \in \mathbb{N}$ such that $\Phi^A_{i,s}(x) \downarrow$
- $\Phi^A_i(x) \uparrow$: there is no $s \in \mathbb{N}$ such that $\Phi^A_{i,s}(x) \downarrow$
- $\phi^A_i(x)$: output of the computation (defined iff $\Phi^A_i(x) \downarrow$)

Introduction
Notation

$\Phi^A_{i,s}(x)$ denotes the **computation** of the TM M_i for $s \in \mathbb{N}$ steps on input $x \in \mathbb{N}$ while having access to an oracle for the subset $A \subseteq \mathbb{N}$

- $\Phi^A_{i,s}(x) \downarrow$: the computation halts after s steps
- $\Phi^A_{i}(x) \downarrow$: there is a $s \in \mathbb{N}$ such that $\Phi^A_{i,s}(x) \downarrow$
- $\Phi^A_{i}(x) \uparrow$: there is no $s \in \mathbb{N}$ such that $\Phi^A_{i,s}(x) \downarrow$
- $\phi^A_{i}(x)$: output of the computation (defined iff $\Phi^A_{i}(x) \downarrow$)

Introduction
Notation

$\Phi^A_{i,s}(x)$ denotes the **computation** of the TM M_i for $s \in \mathbb{N}$ steps on input $x \in \mathbb{N}$ while having access to an oracle for the subset $A \subseteq \mathbb{N}$

- $\Phi^A_{i,s}(x) \downarrow$: the computation halts after s steps
- $\Phi^A_{i}(x) \downarrow$: there is a $s \in \mathbb{N}$ such that $\Phi^A_{i,s}(x) \downarrow$
- $\Phi^A_{i}(x) \uparrow$: there is no $s \in \mathbb{N}$ such that $\Phi^A_{i,s}(x) \downarrow$
- $\phi^A_i(x)$: output of the computation (defined iff $\Phi^A_i(x) \downarrow$)

Introduction

Notation

$\Phi_{i,s}^A(x)$ denotes the **computation** of the TM M_i for $s \in \mathbb{N}$ steps on input $x \in \mathbb{N}$ while having access to an oracle for the subset $A \subseteq \mathbb{N}$

- $\Phi_{i,s}^A(x) \downarrow$: the computation halts after s steps
- $\Phi_i^A(x) \downarrow$: there is a $s \in \mathbb{N}$ such that $\Phi_{i,s}^A(x) \downarrow$
- $\Phi_i^A(x) \uparrow$: there is no $s \in \mathbb{N}$ such that $\Phi_{i,s}^A(x) \downarrow$
- $\phi_i^A(x)$: output of the computation (defined iff $\Phi_i^A(x) \downarrow$)

Introduction
Definitions

Given a set $S \subseteq \mathbb{N}$, we say that S is:

- **Semi-decidable** if $\exists i \in \mathbb{N}$ such that $\forall x \in S$ it holds that $\Phi_i(x) \downarrow$ and $\phi_i(x) = 1$.
- **Decidable** if $\exists i \in \mathbb{N}$ such that $\forall x \in \mathbb{N}$ it holds that $\Phi_i(x) \downarrow$ and $\phi_i(x) = 1$ if $x \in S$, otherwise $\phi_i(x) = 0$.
- **Recursively enumerable** if there is an algorithmic procedure $\mathcal{A} : \mathbb{N} \to \{0, 1\}$ such that $S = \{A(0), A(1), A(2), \ldots\}$

Obs. 1: S is semi-decidable if and only if it is r.e.
Obs. 2: S is decidable if and only if both S and $\overline{S}$ are semi-decidable.

Introduction

Definitions

Given a set $S \subseteq \mathbb{N}$, we say that S is:

- **Semi-decidable** if $\exists i \in \mathbb{N}$ such that $\forall x \in S$ it holds that $\Phi_i(x) \downarrow$ and $\phi_i(x) = 1$.
- **Decidable** if $\exists i \in \mathbb{N}$ such that $\forall x \in \mathbb{N}$ it holds that $\Phi_i(x) \downarrow$ and $\phi_i(x) = 1$ if $x \in S$, otherwise $\phi_i(x) = 0$.
- **Recursively enumerable** if there is an algorithmic procedure $\mathcal{A} : \mathbb{N} \to \{0, 1\}$ such that $S = \{\mathcal{A}(0), \mathcal{A}(1), \mathcal{A}(2), \ldots\}$

Obs. 1: S is semi-decidable if and only if it is r.e.
Obs. 2: S is decidable if and only if both S and $\overline{S}$ are semi-decidable.

Given a set $S \subseteq \mathbb{N}$, we say that S is:

- **Semi-decidable** if $\exists i \in \mathbb{N}$ such that $\forall x \in S$ it holds that $\Phi_i(x) \downarrow$ and $\phi_i(x) = 1$.
- **Decidable** if $\exists i \in \mathbb{N}$ such that $\forall x \in \mathbb{N}$ it holds that $\Phi_i(x) \downarrow$ and $\phi_i(x) = 1$ if $x \in S$, otherwise $\phi_i(x) = 0$.
- **Recursively enumerable** if there is an algorithmic procedure $\mathcal{A} : \mathbb{N} \to \{0, 1\}$ such that $S = \{A(0), A(1), A(2), \ldots\}$

Obs. 1: S is semi-decidable if and only if it is r.e.
Obs. 2: S is decidable if and only if both S and $\overline{S}$ are semi-decidable.

Introduction

Definitions

Given a set $S \subseteq \mathbb{N}$, we say that S is:

- **Semi-decidable** if $\exists i \in \mathbb{N}$ such that $\forall x \in S$ it holds that $\Phi_i(x) \downarrow$ and $\phi_i(x) = 1$.
- **Decidable** if $\exists i \in \mathbb{N}$ such that $\forall x \in \mathbb{N}$ it holds that $\Phi_i(x) \downarrow$ and $\phi_i(x) = 1$ if $x \in S$, otherwise $\phi_i(x) = 0$.
- **Recursively enumerable** if there is an algorithmic procedure $\mathcal{A} : \mathbb{N} \to \{0, 1\}$ such that $S = \{A(0), A(1), A(2), \ldots\}$

Obs. 1: S is semi-decidable if and only if it is r.e.
Obs. 2: S is decidable if and only if both S and $\overline{S}$ are semi-decidable.

Turing [Tur37] proved that:

- Some sets that are semi-decidable but undecidable (e.g. $H = \{(i, x) \mid \Phi_i(x) \downarrow\}$).
- Some sets cannot be semi-decided (e.g. $\overline{H}$).

This gives three degrees of computability: **solvable** problems, **semi-solvable** problems and **unsolvable** problems.

Are there some other degrees of computability?

Turing [Tur37] proved that:

- Some sets that are semi-decidable but undecidable (e.g. $H = \{(i, x) \mid \Phi_i(x) \downarrow\}$).
- Some sets cannot be semi-decided (e.g. $\overline{H}$).

This gives three degrees of computability: **solvable** problems, **semi-solvable** problems and **unsolvable** problems.

Are there some other degrees of computability?

Introduction

Turing's work

Turing [Tur37] proved that:

- Some sets that are semi-decidable but undecidable (e.g. $H = \{(i, x) \mid \Phi_i(x) \downarrow\}$).
- Some sets cannot be semi-decided (e.g. $\overline{H}$).

This gives three degrees of computability: **solvable** problems, **semi-solvable** problems and **unsolvable** problems.

Are there some other degrees of computability?

Introduction
Turing's work

Turing [Tur37] proved that:

- Some sets that are semi-decidable but undecidable (e.g. $H = \{(i, x) \mid \Phi_i(x) \downarrow\}$).
- Some sets cannot be semi-decided (e.g. $\overline{H}$).

This gives three degrees of computability: **solvable** problems, **semi-solvable** problems and **unsolvable** problems.

Are there some other degrees of computability?

Table of Contents

2 Degrees of Unsolvability

- Introduction
- Degrees of Unsolvability
- Post's problem

Degrees of Unsolvability
Turing degrees

Post [Pos44] formalized the idea of computability degrees through **Turing reductions**.

- *Turing reducibility*: $A \leq_T B$ when $\exists i \in \mathbb{N}$ such that $\Phi_i^B(x)\downarrow$ for all $x \in \mathbb{N}$ and $\phi_i^B = A$.
- *Turing equivalence*: $A \equiv_T B$ when $A \equiv_T B$, when $A \leq_T B$ and $B \leq_T A$
- The set $\mathcal{D} = 2^{\mathbb{N}}/_{\equiv_T}$ is referred to as the set of **Turing degrees**.

Post [Pos44] formalized the idea of computability degrees through **Turing reductions**.

- *Turing reducibility*: $A \leq_T B$ when $\exists i \in \mathbb{N}$ such that $\Phi_i^B(x) \downarrow$ for all $x \in \mathbb{N}$ and $\phi_i^B = A$.
- *Turing equivalence*: $A \equiv_T B$ when $A \equiv_T B$, when $A \leq_T B$ and $B \leq_T A$
- The set $\mathcal{D} = 2^{\mathbb{N}}/_{\equiv_T}$ is referred to as the set of **Turing degrees**.

Post [Pos44] formalized the idea of computability degrees through **Turing reductions**.

- *Turing reducibility*: $A \leq_T B$ when $\exists i \in \mathbb{N}$ such that $\Phi_i^B(x) \downarrow$ for all $x \in \mathbb{N}$ and $\phi_i^B = A$.
- *Turing equivalence*: $A \equiv_T B$ when $A \equiv_T B$, when $A \leq_T B$ and $B \leq_T A$
- The set $\mathcal{D} = 2^{\mathbb{N}}/_{\equiv_T}$ is referred to as the set of **Turing degrees**.

Degrees of Unsolvability

Turing degrees

We say that $[A]$ is **lower** than $[B]$, written as $[A] \preceq [B]$, if $A \leq_T B$.

$\preceq$ is a (partial) order over the set $\mathcal{D}$, forming an hierarchy of unsolvability degrees.

Prop. 1: There is an unique degree containing all the decidable problems

- This unique class is referred to as the 0 degree (formally $0 = [\emptyset]$)

Prop. 2: There is no degree below 0

- If $[A] \preceq 0$ then A is decidable, thus $[A] = 0$

Degrees of Unsolvability

Turing degrees

We say that $[A]$ is **lower** than $[B]$, written as $[A] \preceq [B]$, if $A \leq_T B$.

$\preceq$ is a (partial) order over the set $\mathcal{D}$, forming an hierarchy of unsolvability degrees.

Prop. 1: There is an unique degree containing all the decidable problems

- This unique class is referred to as the $\mathbf{0}$ degree (formally $\mathbf{0} = [\emptyset]$)

Prop. 2: There is no degree below $\mathbf{0}$

- If $[A] \preceq \mathbf{0}$ then A is decidable, thus $[A] = \mathbf{0}$

We say that $[A]$ is **lower** than $[B]$, written as $[A] \preceq [B]$, if $A \leq_T B$.

$\preceq$ is a (partial) order over the set $\mathcal{D}$, forming an hierarchy of unsolvability degrees.

Prop. 1: There is an unique degree containing all the decidable problems

- This unique class is referred to as the 0 degree (formally $0 = [\varnothing]$)

Prop. 2: There is no degree below 0

- If $[A] \preceq 0$ then A is decidable, thus $[A] = 0$

We say that $[A]$ is **lower** than $[B]$, written as $[A] \preceq [B]$, if $A \leq_T B$.

$\preceq$ is a (partial) order over the set $\mathcal{D}$, forming an hierarchy of unsolvability degrees.

Prop. 1: There is an unique degree containing all the decidable problems

- This unique class is referred to as the 0 degree (formally $0 = [\varnothing]$)

Prop. 2: There is no degree below 0

- If $[A] \preceq 0$ then A is decidable, thus $[A] = 0$

Degrees of Unsolvability

Turing degrees

We say that $[A]$ is **lower** than $[B]$, written as $[A] \preceq [B]$, if $A \leq_T B$.

$\preceq$ is a (partial) order over the set $\mathcal{D}$, forming an hierarchy of unsolvability degrees.

Prop. 1: There is an unique degree containing all the decidable problems

- This unique class is referred to as the 0 degree (formally $0 = [\varnothing]$)

Prop. 2: There is no degree below 0

- If $[A] \preceq 0$ then A is decidable, thus $[A] = 0$

Degrees of Unsolvability

Turing degrees

We say that $[A]$ is **lower** than $[B]$, written as $[A] \preceq [B]$, if $A \leq_T B$.

$\preceq$ is a (partial) order over the set $\mathcal{D}$, forming an hierarchy of unsolvability degrees.

Prop. 1: There is an unique degree containing all the decidable problems

- This unique class is referred to as the 0 degree (formally $0 = [\varnothing]$)

Prop. 2: There is no degree below 0

- If $[A] \preceq 0$ then A is decidable, thus $[A] = 0$

The strict relation $[A] \prec [B]$ between degrees can be easily forced through **Turing jumps**.

- Given a set $X \subseteq \mathbb{N}$, the *Turing jump* of X is the set $X' = \{i \mid \Phi_i^X(i) \downarrow\}$
  - $X <_T X'$ since X' is obtained by forcing a variant of the Halting problem on TMs with X as an oracle
- **Obs.:** The jump $0'$ of 0 is exactly the class containing the Halting problem

$$\varnothing' = \{i \mid \Phi_i^{\varnothing}(i) \downarrow\} = \{i \mid \Phi_i(i) \downarrow\} \equiv_T H$$

Degrees of Unsolvability

The jump operator

The strict relation $[A] \prec [B]$ between degrees can be easily forced through **Turing jumps**.

- Given a set $X \subseteq \mathbb{N}$, the *Turing jump* of X is the set $X' = \{i \mid \Phi_i^X(i) \downarrow\}$
  - $X <_T X'$ since X' is obtained by forcing a variant of the Halting problem on TMs with X as an oracle
- **Obs.:** The jump $0'$ of 0 is exactly the class containing the Halting problem

$$\varnothing' = \{i \mid \Phi_i^{\varnothing}(i) \downarrow\} = \{i \mid \Phi_i(i) \downarrow\} \equiv_T H$$

Degrees of Unsolvability

The jump operator

The strict relation $[A] \prec [B]$ between degrees can be easily forced through **Turing jumps**.

- Given a set $X \subseteq \mathbb{N}$, the *Turing jump* of X is the set $X' = \{i \mid \Phi_i^X(i) \downarrow\}$
  - $X <_T X'$ since X' is obtained by forcing a variant of the Halting problem on TMs with X as an oracle
- **Obs.:** The jump $0'$ of 0 is exactly the class containing the Halting problem

$$\varnothing' = \{i \mid \Phi_i^{\varnothing}(i) \downarrow\} = \{i \mid \Phi_i(i) \downarrow\} \equiv_T H$$

Degrees of Unsolvability

The jump operator

- **Thm.:** $[A] \leq_m H$ if and only if A is r.e.
  - If $A \leq_m H$ then A is trivially r.e. since H is r.e.
  - If A is r.e. then it has a semi-decider M_i. Let M_j be a new semi-decider such that $\Phi_j(x) \downarrow$ if $\phi_i(x) = 1$, otherwise $\Phi_j(x) \uparrow$.
- **Cor.:** Every degree containing a r.e. set is below $0'$
- **Obs.:** If a degree contains a r.e. set, the other sets aren't forced to also be r.e.
  - E.g.: $\overline{H} \in 0'$, but $\overline{H}$ is not r.e

Degrees of Unsolvability

The jump operator

- **Thm.:** $[A] \leq_m H$ if and only if A is r.e.
  - If $A \leq_m H$ then A is trivially r.e. since H is r.e.
  - If A is r.e. then it has a semi-decider M_i. Let M_j be a new semi-decider such that $\Phi_j(x) \downarrow$ if $\phi_i(x) = 1$, otherwise $\Phi_j(x) \uparrow$.
- **Cor.:** Every degree containing a r.e. set is below $0'$
- **Obs.:** If a degree contains a r.e. set, the other sets <u>aren't</u> forced to also be r.e.
  - E.g.: $\overline{H} \in 0'$, but $\overline{H}$ is not r.e.

Degrees of Unsolvability

The jump operator

- **Thm.:** $[A] \leq_m H$ if and only if A is r.e.
  - If $A \leq_m H$ then A is trivially r.e. since H is r.e.
  - If A is r.e. then it has a semi-decider M_i. Let M_j be a new semi-decider such that $\Phi_j(x) \downarrow$ if $\phi_i(x) = 1$, otherwise $\Phi_j(x) \uparrow$.
- **Cor.:** Every degree containing a r.e. set is below $0'$
- **Obs.:** If a degree contains a r.e. set, the other sets aren't forced to also be r.e.
  - E.g.: $\overline{H} \in 0'$, but $\overline{H}$ is not r.e.

Degrees of Unsolvability
The jump operator

- **Thm.:** $[A] \leq_m H$ if and only if A is r.e.
  - If $A \leq_m H$ then A is trivially r.e. since H is r.e.
  - If A is r.e. then it has a semi-decider M_i. Let M_j be a new semi-decider such that $\Phi_j(x) \downarrow$ if $\phi_i(x) = 1$, otherwise $\Phi_j(x) \uparrow$.
- **Cor.:** Every degree containing a r.e. set is below $0'$
- **Obs.:** If a degree contains a r.e. set, the other sets aren't forced to also be r.e.
  - E.g.: $\overline{H} \in 0'$, but $\overline{H}$ is not r.e

Degrees of Unsolvability

The jump operator

- **Thm.:** $[A] \leq_m H$ if and only if A is r.e.
  - If $A \leq_m H$ then A is trivially r.e. since H is r.e.
  - If A is r.e. then it has a semi-decider M_i. Let M_j be a new semi-decider such that $\Phi_j(x) \downarrow$ if $\phi_i(x) = 1$, otherwise $\Phi_j(x) \uparrow$.
- **Cor.:** Every degree containing a r.e. set is below $0'$
- **Obs.**: If a degree contains a r.e. set, the other sets aren't forced to also be r.e.
  - E.g.: $\overline{H} \in 0'$, but $\overline{H}$ is not r.e

Degrees of Unsolvability

The jump operator

- **Thm.:** $[A] \leq_m H$ if and only if A is r.e.
  - If $A \leq_m H$ then A is trivially r.e. since H is r.e.
  - If A is r.e. then it has a semi-decider M_i. Let M_j be a new semi-decider such that $\Phi_j(x) \downarrow$ if $\phi_i(x) = 1$, otherwise $\Phi_j(x) \uparrow$.
- **Cor.:** Every degree containing a r.e. set is below $0'$
- **Obs.**: If a degree contains a r.e. set, the other sets aren't forced to also be r.e.
  - E.g.: $\overline{H} \in 0'$, but $\overline{H}$ is not r.e

Table of Contents

3 Post's problem

- Introduction
- Degrees of Unsolvability
- Post's problem

Post's problem

The main question

- The jump operator can be iteratively applied to get infinite levels of the hierarchy, i.e. $0 \prec 0' \prec 0'' \prec \ldots$. This makes "going upwards" not so interesting.
- Post [Pos44] proved that for each degree A there is another degree B that is incomparable with A.
  - Cor.: There is at least one Turing degree that is incomparable with both 0 and $0'$
- **Post's problem**: is there a degree d such that $0 \prec d \prec 0'$?

Post's problem

The main question

- The jump operator can be iteratively applied to get infinite levels of the hierarchy, i.e. $0 \prec 0' \prec 0'' \prec \ldots$. This makes "going upwards" not so interesting.
- Post [Pos44] proved that for each degree A there is another degree B that is incomparable with A.
  - **Cor.:** There is at least one Turing degree that is incomparable with both 0 and $0'$
- **Post's problem**: is there a degree d such that $0 \prec d \prec 0'$?

Post's problem

The main question

- The jump operator can be iteratively applied to get infinite levels of the hierarchy, i.e. $0 \prec 0' \prec 0'' \prec \ldots$. This makes "going upwards" not so interesting.
- Post [Pos44] proved that for each degree A there is another degree B that is incomparable with A.
  - **Cor.:** There is at least one Turing degree that is incomparable with both 0 and $0'$
- **Post's problem**: is there a degree d such that $0 \prec d \prec 0'$?

Post's problem

The main question

- The jump operator can be iteratively applied to get infinite levels of the hierarchy, i.e. $0 \prec 0' \prec 0'' \prec \ldots$. This makes "going upwards" not so interesting.
- Post [Pos44] proved that for each degree A there is another degree B that is incomparable with A.
  - **Cor.:** There is at least one Turing degree that is incomparable with both 0 and $0'$
- **Post's problem**: is there a degree d such that $0 \prec d \prec 0'$?

Post's problem

The main question

- Post's problem was solved 12 years later independently by Friedberg [Fri57] and Muchnik [Muc56] through the **finite injury priority method**.
- The method is an improvement on the **finite extension method** developed by Post in his original works
- Since the FIP method involves constructions that are way more complex than those of the FE method, we'll first give an example of the latter

Post's problem

The main question

- Post's problem was solved 12 years later independently by Friedberg [Fri57] and Muchnik [Muc56] through the **finite injury priority method**.
- The method is an improvement on the **finite extension method** developed by Post in his original works
- Since the FIP method involves constructions that are way more complex than those of the FE method, we'll first give an example of the latter

Post's problem

The main question

- Post's problem was solved 12 years later independently by Friedberg [Fri57] and Muchnik [Muc56] through the **finite injury priority method**.
- The method is an improvement on the **finite extension method** developed by Post in his original works
- Since the FIP method involves constructions that are way more complex than those of the FE method, we'll first give an example of the latter

The finite extension method

The setup

- Given a set A, we want to define a countable list of **requirements** $\{R_i\}_{i\in\mathbb{N}}$, each to be satisfied by a string.
- Strings are to be considered as an infinite tape of cells, each marked by an index and containing either a 0, a 1 or an undefined value.
  - In some sense, each string can be viewed as a partial function on $\{0, 1\}$.
- We start with the empty string and on each step $s \in \mathbb{N}$, we construct a new finite string A_{s+1} that extends A_s and satisfies $R_0, R_1, \ldots, R_{s+1}$.

The finite extension method

The setup

- Given a set A, we want to define a countable list of **requirements** $\{R_i\}_{i \in \mathbb{N}}$, each to be satisfied by a string.
- Strings are to be considered as an infinite tape of cells, each marked by an index and containing either a 0, a 1 or an undefined value.
  - In some sense, each string can be viewed as a partial function on $\{0, 1\}$.
- We start with the empty string and on each step $s \in \mathbb{N}$, we construct a new finite string A_{s+1} that extends A_s and satisfies $R_0, R_1, \ldots, R_{s+1}$.

The finite extension method

The setup

- Given a set A, we want to define a countable list of **requirements** $\{R_i\}_{i \in \mathbb{N}}$, each to be satisfied by a string.
- Strings are to be considered as an infinite tape of cells, each marked by an index and containing either a 0, a 1 or an undefined value.
  - In some sense, each string can be viewed as a partial function on $\{0, 1\}$.
- We start with the empty string and on each step $s \in \mathbb{N}$, we construct a new finite string A_{s+1} that extends A_s and satisfies $R_0, R_1, \ldots, R_{s+1}$.

The finite extension method

The setup

- Given a set A, we want to define a countable list of **requirements** $\{R_i\}_{i\in\mathbb{N}}$, each to be satisfied by a string.
- Strings are to be considered as an infinite tape of cells, each marked by an index and containing either a 0, a 1 or an undefined value.
  - In some sense, each string can be viewed as a partial function on $\{0, 1\}$.
- We start with the empty string and on each step $s \in \mathbb{N}$, we construct a new finite string A_{s+1} that extends A_s and satisfies $R_0, R_1, \dots, R_{s+1}$.

[KP54] **Thm.:** There are two sets A and B that are incomparable

- For each $i \in \mathbb{N}$, we define the requirement R_{2i} as $\phi_i^A \neq B$ and R_{2i+1} as $\phi_i^B \neq A$
  - When $\Phi_i^A(x) \uparrow$ or $\Phi_i^B(x) \uparrow$, the requirements R_{2i}, R_{2i+1} are considered to be satisfied.
- Let $\{R_i\}_{i \in \mathbb{N}}$ be the list of requirements
- Let $A_0 = B_0 = \varepsilon$

The finite extension method

The Kleene-Post theorem

[KP54] **Thm.:** There are two sets A and B that are incomparable

- For each $i \in \mathbb{N}$, we define the requirement R_{2i} as $\phi_i^A \neq B$ and R_{2i+1} as $\phi_i^B \neq A$
  - When $\Phi_i^A(x) \uparrow$ or $\Phi_i^B(x) \uparrow$, the requirements R_{2i}, R_{2i+1} are considered to be satisfied.
- Let $\{R_i\}_{i \in \mathbb{N}}$ be the list of requirements
- Let $A_0 = B_0 = \varepsilon$

[KP54] **Thm.:** There are two sets A and B that are incomparable

- For each $i \in \mathbb{N}$, we define the requirement R_{2i} as $\phi_i^A \neq B$ and R_{2i+1} as $\phi_i^B \neq A$
  - When $\Phi_i^A(x)\uparrow$ or $\Phi_i^B(x)\uparrow$, the requirements R_{2i}, R_{2i+1} are considered to be satisfied.
- Let $\{R_i\}_{i\in\mathbb{N}}$ be the list of requirements
- Let $A_0 = B_0 = \varepsilon$

The finite extension method

The Kleene-Post theorem

[KP54] **Thm.:** There are two sets A and B that are incomparable

- For each $i \in \mathbb{N}$, we define the requirement R_{2i} as $\phi_i^A \neq B$ and R_{2i+1} as $\phi_i^B \neq A$
  - When $\Phi_i^A(x)\uparrow$ or $\Phi_i^B(x)\uparrow$, the requirements R_{2i}, R_{2i+1} are considered to be satisfied.
- Let $\{R_i\}_{i\in\mathbb{N}}$ be the list of requirements
- Let $A_0 = B_0 = \varepsilon$

The finite extension method

The Kleene-Post theorem

- Consider a generic step $s \in \mathbb{N}$
  - We assume that $s = 2i$ since the case $2i + 1$ is symmetrically constructed
- Choose any index $x \in \mathbb{N}$ such that $B_s(x) = *$
  - Guaranteed to exist!
- If there is any finite extension A' of A_s such that $\Phi_i^{A'}(x)\downarrow$ then we set:

$$A_{s+1} = A' \qquad B_{s+1}(y) = \begin{cases} B_s(y) & \text{if } y \neq x \\ 1 - \phi_i^{A'}(x) & \text{if } y = x \end{cases}$$

- The requirement R_{2i} is satisfied by this flipped bit.

The finite extension method

The Kleene-Post theorem

- Consider a generic step $s \in \mathbb{N}$
  - We assume that $s = 2i$ since the case $2i + 1$ is symmetrically constructed
- Choose any index $x \in \mathbb{N}$ such that $B_s(x) = *$
  - Guaranteed to exist!
- If there is any finite extension A' of A_s such that $\Phi_i^{A'}(x) \downarrow$ then we set:

$$A_{s+1} = A' \qquad B_{s+1}(y) = \begin{cases} B_s(y) & \text{if } y \neq x \\ 1 - \phi_i^{A'}(x) & \text{if } y = x \end{cases}$$

- The requirement R_{2i} is satisfied by this flipped bit.

The finite extension method

The Kleene-Post theorem

- Consider a generic step $s \in \mathbb{N}$
  - We assume that $s = 2i$ since the case $2i + 1$ is symmetrically constructed
- Choose any index $x \in \mathbb{N}$ such that $B_s(x) = *$
  - Guaranteed to exist!
- If there is any finite extension A' of A_s such that $\Phi_i^{A'}(x) \downarrow$ then we set:

$$A_{s+1} = A' \qquad B_{s+1}(y) = \begin{cases} B_s(y) & \text{if } y \neq x \\ 1 - \phi_i^{A'}(x) & \text{if } y = x \end{cases}$$

- The requirement R_{2i} is satisfied by this flipped bit.

The finite extension method

The Kleene-Post theorem

- Consider a generic step $s \in \mathbb{N}$
  - We assume that $s = 2i$ since the case $2i + 1$ is symmetrically constructed
- Choose any index $x \in \mathbb{N}$ such that $B_s(x) = *$
  - Guaranteed to exist!
- If there is any finite extension A' of A_s such that $\Phi_i^{A'}(x) \downarrow$ then we set:

$$A_{s+1} = A' \qquad B_{s+1}(y) = \begin{cases} B_s(y) & \text{if } y \neq x \\ 1 - \phi_i^{A'}(x) & \text{if } y = x \end{cases}$$

- The requirement R_{2i} is satisfied by this flipped bit.

The finite extension method

The Kleene-Post theorem

- Consider a generic step $s \in \mathbb{N}$
  - We assume that $s = 2i$ since the case $2i + 1$ is symmetrically constructed
- Choose any index $x \in \mathbb{N}$ such that $B_s(x) = *$
  - Guaranteed to exist!
- If there is any finite extension A' of A_s such that $\Phi_i^{A'}(x) \downarrow$ then we set:

$$A_{s+1} = A' \qquad B_{s+1}(y) = \begin{cases} B_s(y) & \text{if } y \neq x \\ 1 - \phi_i^{A'}(x) & \text{if } y = x \end{cases}$$

- The requirement R_{2i} is satisfied by this flipped bit.

The finite extension method

The Kleene-Post theorem

- Consider a generic step $s \in \mathbb{N}$
  - We assume that $s = 2i$ since the case $2i + 1$ is symmetrically constructed
- Choose any index $x \in \mathbb{N}$ such that $B_s(x) = *$
  - Guaranteed to exist!
- If there is any finite extension A' of A_s such that $\Phi_i^{A'}(x) \downarrow$ then we set:

$$A_{s+1} = A' \qquad B_{s+1}(y) = \begin{cases} B_s(y) & \text{if } y \neq x \\ 1 - \phi_i^{A'}(x) & \text{if } y = x \end{cases}$$

- The requirement R_{2i} is satisfied by this flipped bit.

The finite extension method

The Kleene-Post theorem

- If no such finite extension A' of A_s exists, then $\Phi_i^{A'}(x) \uparrow$ for all A'.
- Hence, we can trivially set

$$A_{s+1} = A_s \qquad B_{s+1}(y) = \begin{cases} B_s(y) & \text{if } y \neq x \\ \text{rand}(\{0,1\}) & \text{if } y = x \end{cases}$$

- The requirement R_{2i} is automatically satisfied since $\Phi_i^{A_{s+1}}(x) \uparrow$.
- $A = \bigcup_{i \in \mathbb{N}} A_i$ and $B = \bigcup_{i \in \mathbb{N}} B_i$ are such that $\Phi_s^A(x) \neq B$ and $\Phi_s^B(x) \neq A$ for all $s \in \mathbb{N}$

$\square$

The finite extension method

The Kleene-Post theorem

- If no such finite extension A' of A_s exists, then $\Phi_i^{A'}(x)\uparrow$ for all A'.
- Hence, we can trivially set

$$A_{s+1} = A_s \qquad B_{s+1}(y) = \begin{cases} B_s(y) & \text{if } y \neq x \\ \mathrm{rand}(\{0,1\}) & \text{if } y = x \end{cases}$$

- The requirement R_{2i} is automatically satisfied since $\Phi_i^{A_{s+1}}(x)\uparrow$.
- $A = \bigcup\limits_{i\in\mathbb{N}} A_i$ and $B = \bigcup\limits_{i\in\mathbb{N}} B_i$ are such that $\Phi_s^A(x) \neq B$ and $\Phi_s^B(x) \neq A$ for all $s \in \mathbb{N}$

$\square$

The finite extension method

The Kleene-Post theorem

- If no such finite extension A' of A_s exists, then $\Phi_i^{A'}(x) \uparrow$ for all A'.
- Hence, we can trivially set

$$A_{s+1} = A_s \qquad B_{s+1}(y) = \begin{cases} B_s(y) & \text{if } y \neq x \\ \text{rand}(\{0,1\}) & \text{if } y = x \end{cases}$$

- The requirement R_{2i} is automatically satisfied since $\Phi_i^{A_{s+1}}(x) \uparrow$.
- $A = \bigcup_{i \in \mathbb{N}} A_i$ and $B = \bigcup_{i \in \mathbb{N}} B_i$ are such that $\Phi_s^A(x) \neq B$ and $\Phi_s^B(x) \neq A$ for all $s \in \mathbb{N}$

$\square$

The finite extension method

The Kleene-Post theorem

- If no such finite extension A' of A_s exists, then $\Phi_i^{A'}(x) \uparrow$ for all A'.
- Hence, we can trivially set

$$A_{s+1} = A_s \qquad B_{s+1}(y) = \begin{cases} B_s(y) & \text{if } y \neq x \\ \mathrm{rand}(\{0,1\}) & \text{if } y = x \end{cases}$$

- The requirement R_{2i} is automatically satisfied since $\Phi_i^{A_{s+1}}(x) \uparrow$.
- $A = \bigcup_{i \in \mathbb{N}} A_i$ and $B = \bigcup_{i \in \mathbb{N}} B_i$ are such that $\Phi_s^A(x) \neq B$ and $\Phi_s^B(x) \neq A$ for all $s \in \mathbb{N}$

$\square$

The finite extension method

Strenghts and weaknesses

- We observe that this general method can be easily modified and extended.
  - E.g.: we can force that $[A], [B] \leq 0'$ or even $[A'], [B'] \leq 0'$
- However, the construction used can never guarantee the recursive enumerability of the two sets.
- This problem is fixed by the **finite injury priority method**

The finite extension method

Strenghts and weaknesses

- We observe that this general method can be easily modified and extended.
  - E.g.: we can force that $[A], [B] \leq 0'$ or even $[A'], [B'] \leq 0'$
- However, the construction used can never guarantee the recursive enumerability of the two sets.
- This problem is fixed by the **finite injury priority method**

The finite extension method

Strenghts and weaknesses

- We observe that this general method can be easily modified and extended.
  - E.g.: we can force that $[A], [B] \leq 0'$ or even $[A'], [B'] \leq 0'$
- However, the construction used can never guarantee the recursive enumerability of the two sets.
- This problem is fixed by the **finite injury priority method**

The finite extension method

Strenghts and weaknesses

- We observe that this general method can be easily modified and extended.
  - E.g.: we can force that $[A], [B] \leq 0'$ or even $[A'], [B'] \leq 0'$
- However, the construction used can never guarantee the recursive enumerability of the two sets.
- This problem is fixed by the **finite injury priority method**

The finite injury priority method

The idea

- **Injury**: a requirement gets injured when its satisfaction is not guaranteed anymore.
- The key idea is to assign a priority level to each requirement and force two conditions during the construction of the set A:
    1. Each requirement has finitely many requirements with higher priority
    2. Each requirement can be injured only by requirements with higher priority

The finite injury priority method

The idea

- **Injury**: a requirement gets injured when its satisfaction is not guaranteed anymore.
- The key idea is to assign a priority level to each requirement and force two conditions during the construction of the set A:
    1. Each requirement has finitely many requirements with higher priority
    2. Each requirement can be injured only by requirements with higher priority

The finite injury priority method

The idea

- **Injury**: a requirement gets injured when its satisfaction is not guaranteed anymore.
- The key idea is to assign a priority level to each requirement and force two conditions during the construction of the set A:
    1. Each requirement has finitely many requirements with higher priority
    2. Each requirement can be injured only by requirements with higher priority

The finite injury priority method

The idea

- **Injury**: a requirement gets injured when its satisfaction is not guaranteed anymore.
- The key idea is to assign a priority level to each requirement and force two conditions during the construction of the set A:
    1. Each requirement has finitely many requirements with higher priority
    2. Each requirement can be injured only by requirements with higher priority

The finite injury priority method

The query function

- To keep track of injuries, we'll use the following **query function**
- Let $A \subseteq \mathbb{N}$ and let $i, s, x \in \mathbb{N}$. The query function $\omega_{i,s}^A$ is defined as:

$$\omega_{i,s}^A(x) = \begin{cases} \max\{z \in \mathbb{N} \mid A(z) \text{ is queried in } \Phi_{i,s}^A\} & \text{if } \Phi_{i,s}^A(x) \downarrow \\ -1 & \text{otherwise} \end{cases}$$

The finite injury priority method

The query function

- To keep track of injuries, we'll use the following **query function**
- Let $A \subseteq \mathbb{N}$ and let $i, s, x \in \mathbb{N}$. The query function $\omega_{i,s}^A$ is defined as:

$$\omega_{i,s}^A(x) = \begin{cases} \max\{z \in \mathbb{N} \mid A(z) \text{ is queried in } \Phi_{i,s}^A\} & \text{if } \Phi_{i,s}^A(x) \downarrow \\ -1 & \text{otherwise} \end{cases}$$

[Fri57; Muc56] **Thm.:** There are two r.e. sets A and B that are incomparable

- The requirements are defined as in the previous theorem
- $A_0, A_1, A_2, \ldots$ and $B_0, B_1, B_2, \ldots$ are now **sets**
- We say that an index x is a **witness** for the requirement R_{2i} at step s if $\Phi_{i,s}^{B_s}(x) \downarrow$ and $\phi_{i,s}^{B_s}(x) \neq A_s(x)$ (symmetric definition for R_{2i+1})

[Fri57; Muc56] **Thm.:** There are two r.e. sets A and B that are incomparable

- The requirements are defined as in the previous theorem
- $A_0, A_1, A_2, \ldots$ and $B_0, B_1, B_2, \ldots$ are now **sets**
- We say that an index x is a **witness** for the requirement R_{2i} at step s if $\Phi_{i,s}^{B_s}(x) \downarrow$ and $\phi_{i,s}^{B_s}(x) \neq A_s(x)$ (symmetric definition for R_{2i+1})

The finite injury priority method

The Friedberg-Muchnik theorem

[Fri57; Muc56] **Thm.:** There are two r.e. sets A and B that are incomparable

- The requirements are defined as in the previous theorem
- $A_0, A_1, A_2, \ldots$ and $B_0, B_1, B_2, \ldots$ are now **sets**
- We say that an index x is a **witness** for the requirement R_{2i} at step s if $\Phi_{i,s}^{B_s}(x)\downarrow$ and $\phi_{i,s}^{B_s}(x) \neq A_s(x)$ (symmetric definition for R_{2i+1})

The finite injury priority method

The Friedberg-Muchnik theorem

- On each step $s \in \mathbb{N}$, for each $j \in \mathbb{N}$ we compute:
  - A witness $w_{j,s}$ for R_j at step s
  - A restriction index $r_{j,s}$ to dictate the priority of the requirements
- We'll enforce that $w_{j,s} = r_{j,s} = -1$ holds when no witness is known for R_j at step s
- The indices that come before each $r_{j,s}$ are considered to be *safe*, i.e. they don't break any requirement.
- R_j gets *injured* at step s if $A_{s+1} - A_s$ contains some $x \leq r_{j,s}$.

The finite injury priority method

The Friedberg-Muchnik theorem

- On each step $s \in \mathbb{N}$, for each $j \in \mathbb{N}$ we compute:
  - A witness $w_{j,s}$ for R_j at step s
  - A restriction index $r_{j,s}$ to dictate the priority of the requirements
- We'll enforce that $w_{j,s} = r_{j,s} = -1$ holds when no witness is known for R_j at step s
- The indices that come before each $r_{j,s}$ are considered to be *safe*, i.e. they don't break any requirement.
- R_j gets *injured* at step s if $A_{s+1} - A_s$ contains some $x \leq r_{j,s}$.

The finite injury priority method

The Friedberg-Muchnik theorem

- On each step $s \in \mathbb{N}$, for each $j \in \mathbb{N}$ we compute:
  - A witness $w_{j,s}$ for R_j at step s
  - A restriction index $r_{j,s}$ to dictate the priority of the requirements
- We'll enforce that $w_{j,s} = r_{j,s} = -1$ holds when no witness is known for R_j at step s
- The indices that come before each $r_{j,s}$ are considered to be *safe*, i.e. they don't break any requirement.
- R_j gets *injured* at step s if $A_{s+1} - A_s$ contains some $x \leq r_{j,s}$.

The finite injury priority method

The Friedberg-Muchnik theorem

- On each step $s \in \mathbb{N}$, for each $j \in \mathbb{N}$ we compute:
  - A witness $w_{j,s}$ for R_j at step s
  - A restriction index $r_{j,s}$ to dictate the priority of the requirements
- We'll enforce that $w_{j,s} = r_{j,s} = -1$ holds when no witness is known for R_j at step s
- The indices that come before each $r_{j,s}$ are considered to be *safe*, i.e. they don't break any requirement.
- R_j gets *injured* at step s if $A_{s+1} - A_s$ contains some $x \leq r_{j,s}$.

The finite injury priority method

The Friedberg-Muchnik theorem

- On each step $s \in \mathbb{N}$, for each $j \in \mathbb{N}$ we compute:
  - A witness $w_{j,s}$ for R_j at step s
  - A restriction index $r_{j,s}$ to dictate the priority of the requirements
- We'll enforce that $w_{j,s} = r_{j,s} = -1$ holds when no witness is known for R_j at step s
- The indices that come before each $r_{j,s}$ are considered to be *safe*, i.e. they don't break any requirement.
- R_j gets *injured* at step s if $A_{s+1} - A_s$ contains some $x \leq r_{j,s}$.

The finite injury priority method

The Friedberg-Muchnik theorem

- On each step $s \in \mathbb{N}$, for each $j \in \mathbb{N}$ we compute:
  - A witness $w_{j,s}$ for R_j at step s
  - A restriction index $r_{j,s}$ to dictate the priority of the requirements
- We'll enforce that $w_{j,s} = r_{j,s} = -1$ holds when no witness is known for R_j at step s
- The indices that come before each $r_{j,s}$ are considered to be *safe*, i.e. they don't break any requirement.
- R_j gets *injured* at step s if $A_{s+1} - A_s$ contains some $x \leq r_{j,s}$.

The finite injury priority method

The Friedberg-Muchnik theorem

- Let $A_0 = B_0 = \varnothing$ and let $w_{j,0} = r_{j,0} = -1$ for each $j \in \mathbb{N}$.
- Consider a generic step $s \in \mathbb{N}$.
  - Assume $s = 2i$.
- If $w_{2i,s} \neq -1$ we propagate the previous values because R_{2i} already has a witness.
  - We set $A_{s+1} = A_s$, $B_{s+1} = B_s$, $w_{2i,s+1} = w_{2i,s}$ and $r_{2i,s+1} = r_{2i,s}$

The finite injury priority method

The Friedberg-Muchnik theorem

- Let $A_0 = B_0 = \varnothing$ and let $w_{j,0} = r_{j,0} = -1$ for each $j \in \mathbb{N}$.
- Consider a generic step $s \in \mathbb{N}$.
  - Assume $s = 2i$.
- If $w_{2i,s} \neq -1$ we propagate the previous values because R_{2i} already has a witness.
  - We set $A_{s+1} = A_s$, $B_{s+1} = B_s$, $w_{2i,s+1} = w_{2i,s}$ and $r_{2i,s+1} = r_{2i,s}$

The finite injury priority method

The Friedberg-Muchnik theorem

- Let $A_0 = B_0 = \varnothing$ and let $w_{j,0} = r_{j,0} = -1$ for each $j \in \mathbb{N}$.
- Consider a generic step $s \in \mathbb{N}$.
  - Assume $s = 2i$.
- If $w_{2i,s} \neq -1$ we propagate the previous values because R_{2i} already has a witness.
  - We set $A_{s+1} = A_s$, $B_{s+1} = B_s$, $w_{2i,s+1} = w_{2i,s}$ and $r_{2i,s+1} = r_{2i,s}$

The finite injury priority method

The Friedberg-Muchnik theorem

- Let $A_0 = B_0 = \varnothing$ and let $w_{j,0} = r_{j,0} = -1$ for each $j \in \mathbb{N}$.
- Consider a generic step $s \in \mathbb{N}$.
  - Assume $s = 2i$.
- If $w_{2i,s} \neq -1$ we propagate the previous values because R_{2i} already has a witness.
  - We set $A_{s+1} = A_s$, $B_{s+1} = B_s$, $w_{2i,s+1} = w_{2i,s}$ and $r_{2i,s+1} = r_{2i,s}$

The finite injury priority method

The Friedberg-Muchnik theorem

- Let $A_0 = B_0 = \varnothing$ and let $w_{j,0} = r_{j,0} = -1$ for each $j \in \mathbb{N}$.
- Consider a generic step $s \in \mathbb{N}$.
  - Assume $s = 2i$.
- If $w_{2i,s} \neq -1$ we propagate the previous values because R_{2i} already has a witness.
  - We set $A_{s+1} = A_s$, $B_{s+1} = B_s$, $w_{2i,s+1} = w_{2i,s}$ and $r_{2i,s+1} = r_{2i,s}$

The finite injury priority method

The Friedberg-Muchnik theorem

- Assume that $w_{2i,s} = -1$. Let x be the minimum index such that $x \notin A_s$ and such that for all $k < 2i$ it holds that $x > r_{k,s}$.
- If $\Phi_{i,s}^{B_s}(x) \uparrow$, we preserve that $w_{2i,s+1} = r_{2i,s+1} = -1$ and propagate $A_{s+1} = A_s$, $B_{s+1} = B_s$. This trivially satisfies the requirement R_{2i}.
- Otherwise, if $\Phi_{i,s}^{B_s}(x) \downarrow$, we set:

$$w_{2i,s+1} = x \qquad r_{2i,s+1} = \max(x, \omega_{i,s}^{B_s}(x))$$

$$A_{s+1} = A_s \cup \{x\} \qquad B_{s+1} = B_s$$

The finite injury priority method

The Friedberg-Muchnik theorem

- Assume that $w_{2i,s} = -1$. Let x be the minimum index such that $x \notin A_s$ and such that for all $k < 2i$ it holds that $x > r_{k,s}$.
- If $\Phi_{i,s}^{B_s}(x) \uparrow$, we preserve that $w_{2i,s+1} = r_{2i,s+1} = -1$ and propagate $A_{s+1} = A_s$, $B_{s+1} = B_s$. This trivially satisfies the requirement R_{2i}.
- Otherwise, if $\Phi_{i,s}^{B_s}(x) \downarrow$, we set:

$$w_{2i,s+1} = x \qquad r_{2i,s+1} = \max(x, \omega_{i,s}^{B_s}(x))$$

$$A_{s+1} = A_s \cup \{x\} \qquad B_{s+1} = B_s$$

The finite injury priority method

The Friedberg-Muchnik theorem

- Assume that $w_{2i,s} = -1$. Let x be the minimum index such that $x \notin A_s$ and such that for all $k < 2i$ it holds that $x > r_{k,s}$.
- If $\Phi_{i,s}^{B_s}(x) \uparrow$, we preserve that $w_{2i,s+1} = r_{2i,s+1} = -1$ and propagate $A_{s+1} = A_s$, $B_{s+1} = B_s$. This trivially satisfies the requirement R_{2i}.
- Otherwise, if $\Phi_{i,s}^{B_s}(x) \downarrow$, we set:

$$w_{2i,s+1} = x \qquad r_{2i,s+1} = \max(x, \omega_{i,s}^{B_s}(x))$$

$$A_{s+1} = A_s \cup \{x\} \qquad B_{s+1} = B_s$$

The finite injury priority method

The Friedberg-Muchnik theorem

Claim: For each $j \in \mathbb{N}$ it holds that:

1. R_j is injured a finite number of times
2. There is a step s_0 such that for all $s \geq s_0$ and for all $k < j$ no new index is added to A_s

Proof.

- The requirement R_j is injured at step s if some $x \leq r_{j,s}$ is added to A_s, which happens only when there is a $k < j$ that adds a new index to A_s, i.e. when

$$w_{j,k} = -1 \quad \Phi_{j,s}^{B_s}(x) \downarrow \quad \phi_{j,s}^{B_s}(x) = 0$$

- In other words, statement (2) of the claim follows from statement (1) by construction.

The finite injury priority method

The Friedberg-Muchnik theorem

Claim: For each $j \in \mathbb{N}$ it holds that:

1. R_j is injured a finite number of times
2. There is a step s_0 such that for all $s \geq s_0$ and for all $k < j$ no new index is added to A_s

Proof.

- The requirement R_j is injured at step s if some $x \leq r_{j,s}$ is added to A_s, which happens only when there is a $k < j$ that adds a new index to A_s, i.e. when

$$w_{j,k} = -1 \quad \Phi_{j,s}^{B_s}(x) \downarrow \quad \phi_{j,s}^{B_s}(x) = 0$$

- In other words, statement (2) of the claim follows from statement (1) by construction.

- Let $g_{j,s} = \sum_{\substack{k<j \text{ s.t.} \\ w_{k,s}\neq -1}} 2^{-(k+1)}$ be the injure value of R_j at step s
- When R_j is injured at step s, the smallest value $k < j$ such that $w_{k,s} \neq w_{k,s+1}$ must satisfy $w_{k,s} = -1$ and $w_{k,s+1} \neq -1$ by construction. Thus:

$$g_{j,s+1} - g_{j,s} \geq 2^{-(j+1)} - \sum_{k<h<j} 2^{-(h+1)} = 2^{-j}$$

- Hence, for each step s we have that $0 \leq g_{j,s} \leq 1 - 2^{-j}$, concluding that R_j can be injured at most $2^j - 1$ times

The finite injury priority method

The Friedberg-Muchnik theorem

- Let $g_{j,s} = \sum_{\substack{k<j \ s.t. \\ w_{k,s}\neq -1}} 2^{-(k+1)}$ be the injure value of R_j at step s
- When R_j is injured at step s, the smallest value $k < j$ such that $w_{k,s} \neq w_{k,s+1}$ must satisfy $w_{k,s} = -1$ and $w_{k,s+1} \neq -1$ by construction. Thus:

$$g_{j,s+1} - g_{j,s} \geq 2^{-(j+1)} - \sum_{k<h<j} 2^{-(h+1)} = 2^{-j}$$

- Hence, for each step s we have that $0 \leq g_{j,s} \leq 1 - 2^{-j}$, concluding that R_j can be injured at most $2^j - 1$ times

The finite injury priority method

The Friedberg-Muchnik theorem

- Let $g_{j,s} = \sum_{\substack{k<j \text{ s.t.} \\ w_{k,s} \neq -1}} 2^{-(k+1)}$ be the injure value of R_j at step s
- When R_j is injured at step s, the smallest value $k < j$ such that $w_{k,s} \neq w_{k,s+1}$ must satisfy $w_{k,s} = -1$ and $w_{k,s+1} \neq -1$ by construction. Thus:

$$g_{j,s+1} - g_{j,s} \geq 2^{-(j+1)} - \sum_{k<h<j} 2^{-(h+1)} = 2^{-j}$$

- Hence, for each step s we have that $0 \leq g_{j,s} \leq 1 - 2^{-j}$, concluding that R_j can be injured at most $2^j - 1$ times

The finite injury priority method

The Friedberg-Muchnik theorem

- We observe that there is always a large enough step s_0^* satisfying statements (1) and (2) of the Claim.
- This step guarantees that for each $j \in \mathbb{N}$ it holds that $\lim_{s \to +\infty} w_{j,s}$ and $\lim_{s \to +\infty} r_{j,s}$ exist and are finite.
  - If for some $s' \geq s_0^*$ it holds that $w_{j,s'} \neq -1$ then it will hold from s' onward
  - Otherwise, there is a minimum value x_s^* such that $x_s^* \notin A_s$ and such that $x_s^* > r_{k,s}$ for all $k < j$
  - By construction $\Phi_{j,s}^{B_s}(x) \uparrow$ holds in this case

The finite injury priority method

The Friedberg-Muchnik theorem

- This concludes that, eventually, each requirement R_j will be satisfied by the construction.
- Thus, $A = \bigcup_{i \in \mathbb{N}} A_i$ and $B = \bigcup_{i \in \mathbb{N}} B_i$ are such that $\Phi_s^A(x) \neq B$ and $\Phi_s^B(x) \neq A$ for all $s \in \mathbb{N}$
- Moreover, the use of the restriction values $r_{j,s}$ allows us to recursively enumerate the sets A and B by restricting our interest to the indexes between 0 and $r_{j,s}$ for each $j \in \mathbb{N}$
  - This implies that A and B are both r.e.!

$\square$

The finite injury priority method

The Friedberg-Muchnik theorem

- This concluses that, eventually, each requirement R_j will be satisfied by the construction.
- Thus, $A = \bigcup_{i\in\mathbb{N}} A_i$ and $B = \bigcup_{i\in\mathbb{N}} B_i$ are such that $\Phi_s^A(x) \neq B$ and $\Phi_s^B(x) \neq A$ for all $s \in \mathbb{N}$
- Moreover, the use of the restriction values $r_{j,s}$ allows us to recursively enumerate the sets A and B by restricting our interest to the indexes between 0 and $r_{j,s}$ for each $j \in \mathbb{N}$
  - This implies that A and B are both r.e.!

$\square$

The finite injury priority method

The Friedberg-Muchnik theorem

- This concludes that, eventually, each requirement R_j will be satisfied by the construction.
- Thus, $A = \bigcup\limits_{i \in \mathbb{N}} A_i$ and $B = \bigcup\limits_{i \in \mathbb{N}} B_i$ are such that $\Phi_s^A(x) \neq B$ and $\Phi_s^B(x) \neq A$ for all $s \in \mathbb{N}$
- Moreover, the use of the restriction values $r_{j,s}$ allows us to recursively enumerate the sets A and B by restricting our interest to the indexes between 0 and $r_{j,s}$ for each $j \in \mathbb{N}$
  - This implies that A and B are both r.e.!

$\square$

- This concludes that, eventually, each requirement R_j will be satisfied by the construction.
- Thus, $A = \bigcup_{i \in \mathbb{N}} A_i$ and $B = \bigcup_{i \in \mathbb{N}} B_i$ are such that $\Phi_s^A(x) \neq B$ and $\Phi_s^B(x) \neq A$ for all $s \in \mathbb{N}$
- Moreover, the use of the restriction values $r_{j,s}$ allows us to recursively enumerate the sets A and B by restricting our interest to the indexes between 0 and $r_{j,s}$ for each $j \in \mathbb{N}$
  - This implies that A and B are both r.e.!

$\square$

- **Infinite Injury Priority Argument**: Sacks [Sac64] proved that the above construction can be extended to a countably infinite argument
- **Density of r.e. sets**: For each pair of r.e. sets A, B there is another r.e. set such that $A <_T C <_T B$
- Simpson [Sim77] proved that the first-order theory of $\mathcal{D}$ over the language $(\preceq, =)$ is many-one equivalent to the theory of true second-order arithmetic.

The finite injury priority method

Other results

- **Infinite Injury Priority Argument**: Sacks [Sac64] proved that the above construction can be extended to a countably infinite argument
- **Density of r.e. sets**: For each pair of r.e. sets A, B there is another r.e. set such that $A <_T C <_T B$
- Simpson [Sim77] proved that the first-order theory of $\mathcal{D}$ over the language $(\preceq, =)$ is many-one equivalent to the theory of true second-order arithmetic.

The finite injury priority method

Other results

- **Infinite Injury Priority Argument**: Sacks [Sac64] proved that the above construction can be extended to a countably infinite argument
- **Density of r.e. sets**: For each pair of r.e. sets A, B there is another r.e. set such that $A <_T C <_T B$
- Simpson [Sim77] proved that the first-order theory of $\mathcal{D}$ over the language $(\preceq, =)$ is many-one equivalent to the theory of true second-order arithmetic.

Thank you for listening!
Any questions?

Bibliography

[Fri57] Richard M. Friedberg. “Two recursively enumerable sets of incomparable degrees of unsolvability (solution of Post’s problem, 1944)”. In: (1957).

[KP54] Stephen C. Kleene and Emil L. Post. “The upper semi-lattice of degrees of recursive unsolvability”. In: (1954).

[Muc56] Albert A. Muchnik. “On the unsolvability of the problem of reducibility in the theory of algorithms”. In: (1956).

[Pos44] Emil L. Post. “Recursively enumerable sets of positive integers and their decision problems”. In: (1944).

[Sac64] Gerald E. Sacks. “The Recursively Enumerable Degrees are Dense”. In: (1964).

[Sim77] Stephen G. Simpson. “First-Order Theory of the Degrees of Recursive Unsolvability”. In: (1977).

[Tur37] Alan M. Turing. “On Computable Numbers, with an Application to the Entscheidungsproblem”. In: (1937).