Algorithms and Computability

Lecture 10: Reductions

Christian Schilling (christianms@cs.aau.dk)

slides courtesy of Martin Zimmermann

Last Time in Algorithms and Computability

We have seen

- Diagonalization
- Non-computability of the halting problem
- Closure properties of computable and computably-enumerable

Encoding of Turing Machines

Let $M = (Q, \Sigma, \Gamma, s, t, r, \delta)$ be a DTM. We assume without loss of generality that $Q = \{1, 11, \ldots, 1^{|Q|}\}$ and $\Sigma = \{0, 1\}$.
Also, let $rep_\Gamma : \Gamma \to \{1, 11, \ldots, 1^{|\Gamma|}\}$ be an encoding of Γ such that $rep_\Gamma(0) = 1$, $rep_\Gamma(1) = 11$, and $rep_\Gamma(\text{␣}) = 111$

Then, M is encoded by the word $\ulcorner M \urcorner$ over $\{0, 1\}$ defined as follows:

$$1^{|Q|}\,0\,1^{|\Gamma|}\,0\,s\,0\,t\,0\,r\,0\,w_\delta$$

where w_δ is the list of encodings of transitions.
Each $\delta(q, b) = (p, a, d)$ is encoded by

$$q\,0\,rep_\Gamma(b)\,0\,p\,0\,rep_\Gamma(a)\,0\,dir(d)\,0$$

where $dir(-1) = 1$ and $dir(+1) = 11$

The Halting Problem

- So, (the encoding of) a DTM can be the input for a DTM
- Thus, we can formulate decision problems about DTMs!
- A particular interesting (both practically and theoretically) problem is the **halting problem**:

$$\mathrm{HP} = \{\langle \ulcorner M \urcorner, w \rangle \mid M \text{ is a DTM that halts on input } w\}$$

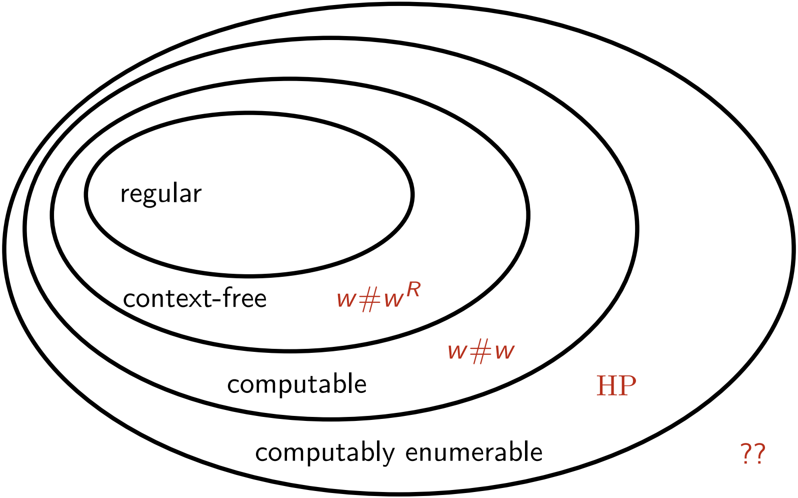
- Here, $\langle \cdot, \cdot \rangle$ is a function that encodes two words x, y over $\{0, 1\}$ by a single word $\langle x, y \rangle$ over $\{0, 1\}$. See "Encoding pairs" in "Computability and Complexity" (page 89) for details

Theorem (Turing 1936)

The halting problem is not computable

Today

- How to show more problems non-computable?
- How to show that a problem is not computably-enumerable?



Agenda

1. **Intuition Behind Reductions**
2. Reductions
3. Applications
4. More Non-computable Problems

The Acceptance Problem for Turing Machines

$$\mathrm{AP} = \{\langle \ulcorner M \urcorner, w\rangle \mid w \in L(M)\}$$

The Acceptance Problem for Turing Machines

$$\mathrm{AP} = \{\langle \ulcorner M \urcorner, w\rangle \mid w \in L(M)\}$$

Given $\ulcorner M \urcorner$, let $f(\ulcorner M \urcorner) = \ulcorner M' \urcorner$, where M' is the DTM obtained from M by replacing every occurrence of r in a transition by t

The Acceptance Problem for Turing Machines

$$\mathrm{AP} = \{\langle \ulcorner M \urcorner, w\rangle \mid w \in L(M)\}$$

Given $\ulcorner M \urcorner$, let $f(\ulcorner M \urcorner) = \ulcorner M' \urcorner$, where M' is the DTM obtained from M by replacing every occurrence of r in a transition by t

- If M accepts w, then M' accepts w
- If M rejects w, then M' accepts w
- If M does not halt on w, then M' does not either

The Acceptance Problem for Turing Machines

$$\mathrm{AP} = \{\langle \ulcorner M \urcorner, w\rangle \mid w \in L(M)\}$$

Given $\ulcorner M \urcorner$, let $f(\ulcorner M \urcorner) = \ulcorner M' \urcorner$, where M' is the DTM obtained from M by replacing every occurrence of r in a transition by t

$$\langle \ulcorner M \urcorner, w\rangle \in \mathrm{HP} \quad \Leftrightarrow \quad \langle f(\ulcorner M \urcorner), w\rangle \in \mathrm{AP}$$

- Can AP be computable?

The Acceptance Problem for Turing Machines

$$\mathrm{AP} = \{\langle \ulcorner M \urcorner, w \rangle \mid w \in L(M)\}$$

Given $\ulcorner M \urcorner$, let $f(\ulcorner M \urcorner) = \ulcorner M' \urcorner$, where M' is the DTM obtained from M by replacing every occurrence of r in a transition by t

$$\langle \ulcorner M \urcorner, w \rangle \in \mathrm{HP} \quad \Leftrightarrow \quad \langle f(\ulcorner M \urcorner), w \rangle \in \mathrm{AP}$$

- Assume there is a halting DTM A for AP
- Then, we can construct a DTM H that computes HP:
  1. Given input $\langle \ulcorner M \urcorner, w \rangle$, write $\langle f(\ulcorner M \urcorner), w \rangle$ on tape
  2. Simulate A on that input (which will halt by assumption)
  3. If A accepts, H accepts as well
  4. If A rejects, H rejects as well
- However, H cannot exist, since HP is not computable.
  Thus, A cannot exist and AP cannot be computable either

The Other Way Around

$$\mathrm{AP} = \{\langle \ulcorner M \urcorner, w \rangle \mid w \in L(M)\}$$

The Other Way Around

$$\mathrm{AP} = \{\langle \ulcorner M \urcorner, w\rangle \mid w \in L(M)\}$$

Given $\ulcorner M \urcorner$, let $f(\ulcorner M \urcorner) = \ulcorner M' \urcorner$, where M' is the DTM obtained from M by replacing every occurrence of r in a transition by a looping state (and adding relevant transitions)

The Other Way Around

$$\mathrm{AP} = \{\langle \ulcorner M \urcorner, w \rangle \mid w \in L(M)\}$$

Given $\ulcorner M \urcorner$, let $f(\ulcorner M \urcorner) = \ulcorner M' \urcorner$, where M' is the DTM obtained from M by replacing every occurrence of r in a transition by a looping state (and adding relevant transitions)

- If M accepts w, then M' accepts w
- If M rejects w, then M' does not halt on w
- If M does not halt on w, then M' does not either

The Other Way Around

$$\mathrm{AP} = \{\langle \ulcorner M \urcorner, w \rangle \mid w \in L(M)\}$$

Given $\ulcorner M \urcorner$, let $f(\ulcorner M \urcorner) = \ulcorner M' \urcorner$, where M' is the DTM obtained from M by replacing every occurrence of r in a transition by a looping state (and adding relevant transitions)

$$\langle \ulcorner M \urcorner, w \rangle \in \mathrm{AP} \quad \Leftrightarrow \quad \langle f(\ulcorner M \urcorner), w \rangle \in \mathrm{HP}$$

The Other Way Around

$$\mathrm{AP} = \{\langle \ulcorner M \urcorner, w\rangle \mid w \in L(M)\}$$

Given $\ulcorner M \urcorner$, let $f(\ulcorner M \urcorner) = \ulcorner M' \urcorner$, where M' is the DTM obtained from M by replacing every occurrence of r in a transition by a looping state (and adding relevant transitions)

$$\langle \ulcorner M \urcorner, w\rangle \in \mathrm{AP} \quad \Leftrightarrow \quad \langle f(\ulcorner M \urcorner), w\rangle \in \mathrm{HP}$$

- There is a DTM H with $L(H) = \mathrm{HP}$ (see Exercise Sheet 3)
- We can construct a DTM A with $L(A) = \mathrm{AP}$:
  1. Given input $\langle \ulcorner M \urcorner, w\rangle$, write $\langle f(\ulcorner M \urcorner), w\rangle$ on tape
  2. Simulate H on that input
  3. If H accepts, A accepts as well
  4. If H rejects, A rejects as well
- H halts on all accepted inputs, and hence so does A
- Thus, AP is computably-enumerable as well

The Usefulness of Reductions

In general, we have reduced instances of a problem A to instances of another problem B (using a function)

- If B is computable, then A is also computable
- If B is computably-enumerable, then A is also computably-enumerable

The Usefulness of Reductions

In general, we have reduced instances of a problem A to instances of another problem B (using a function)

- If B is computable, then A is also computable
- If B is computably-enumerable, then A is also computably-enumerable

The converse is also very useful:

- If A is not computable, then B is also not computable
- If A is not computably-enumerable, then B is also not computably-enumerable

The Usefulness of Reductions

In general, we have reduced instances of a problem A to instances of another problem B (using a function)

- If B is computable, then A is also computable
- If B is computably-enumerable, then A is also computably-enumerable

The converse is also very useful:

- If A is not computable, then B is also not computable
- If A is not computably-enumerable, then B is also not computably-enumerable

In the following, we formalize the notion of "reduction" and show some applications

The Usefulness of Reductions

In general, we have reduced instances of a problem A to instances of another problem B (using a function)

- If B is computable, then A is also computable
- If B is computably-enumerable, then A is also computably-enumerable

The converse is also very useful:

- If A is not computable, then B is also not computable
- If A is not computably-enumerable, then B is also not computably-enumerable

In the following, we formalize the notion of "reduction" and show some applications

Note

There are different notions of reductions for different applications. We focus here on one notion that is useful for our goals

Agenda

Caution

Let us "reduce" the halting problem

$$\mathrm{HP} = \{\langle \ulcorner M \urcorner, w\rangle \mid M \text{ is a DTM that halts on input } w\}$$

to the computable language $O = \{1\} \subseteq \mathbb{B}^*$ by defining the following function f:

- If $w \in \mathrm{HP}$, then define $f(w) = 1 \in O$
- If $w \notin \mathrm{HP}$, then define $f(w) = 0 \notin O$

Caution

Let us "reduce" the halting problem

$$\mathrm{HP} = \{\langle \ulcorner M \urcorner, w\rangle \mid M \text{ is a DTM that halts on input } w\}$$

to the computable language $O = \{1\} \subseteq \mathbb{B}^*$ by defining the following function f:

- If $w \in \mathrm{HP}$, then define $f(w) = 1 \in O$
- If $w \notin \mathrm{HP}$, then define $f(w) = 0 \notin O$

A problem:

- We can now compute whether a given input w is in HP by computing $f(w)$ and checking whether it is in O or not
- But HP is not computable! Where is the issue?

Caution

Let us "reduce" the halting problem

$$\mathrm{HP} = \{\langle \ulcorner M \urcorner, w \rangle \mid M \text{ is a DTM that halts on input } w\}$$

to the computable language $O = \{1\} \subseteq \mathbb{B}^*$ by defining the following function f:

- If $w \in \mathrm{HP}$, then define $f(w) = 1 \in O$
- If $w \notin \mathrm{HP}$, then define $f(w) = 0 \notin O$

A problem:

- We can now compute whether a given input w is in HP by computing $f(w)$ and checking whether it is in O or not
- But HP is not computable! Where is the issue?

  The case distinction solves a non-computable problem!

Definition (See Def. 2.1.17 in "Computability and Complexity" for full definition)

A function $f\colon \Sigma_1^* \to \Sigma_2^*$ is a computable function if and only if there exists a DTM M_f that on every input $w \in \Sigma_1^*$

- always halts and accepts
- with just $f(w)$ on its tape and
- the head at the first letter of $f(w)$

Computable Functions

Definition (See Def. 2.1.17 in "Computability and Complexity" for full definition)

A function $f \colon \Sigma_1^* \to \Sigma_2^*$ is a computable function if and only if there exists a DTM M_f that on every input $w \in \Sigma_1^*$

- always halts and accepts
- with just $f(w)$ on its tape and
- the head at the first letter of $f(w)$

Examples

- $f(w) = ww$ is computable
- $f(\ulcorner m \urcorner \# \ulcorner n \urcorner) = \ulcorner m \cdot n \urcorner$ is computable (where $\ulcorner n \urcorner$ denotes the binary encoding of $n \in \mathbb{N}$)

A More Interesting Example

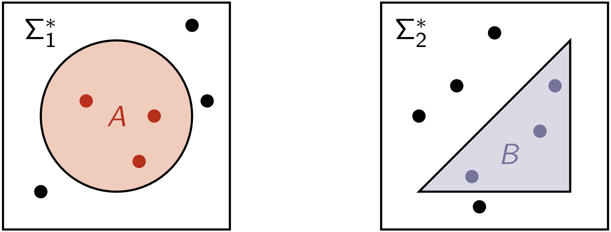
$$f(w) = \begin{cases} \ulcorner M' \urcorner & \text{if } w = \ulcorner M \urcorner \text{ for some Turing machine } M \\ & \text{and } M' \text{ is the Turing machine obtained} \\ & \text{from } M \text{ by replacing every occurrence of its} \\ & \text{rejecting state in a transition by the accepting one} \\ \varepsilon & \text{if } w \text{ is not an encoding of a Turing machine} \end{cases}$$

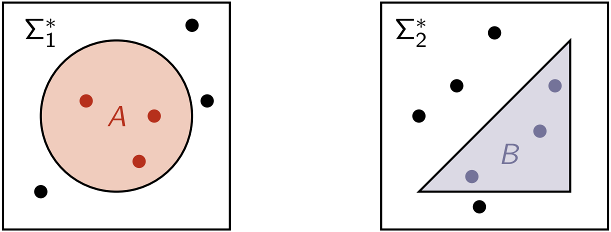
A More Interesting Example

$$f(w) = \begin{cases} \ulcorner M' \urcorner & \text{if } w = \ulcorner M \urcorner \text{ for some Turing machine } M \\ & \text{and } M' \text{ is the Turing machine obtained} \\ & \text{from } M \text{ by replacing every occurrence of its} \\ & \text{rejecting state in a transition by the accepting one} \\ \varepsilon & \text{if } w \text{ is not an encoding of a Turing machine} \end{cases}$$

f is computed by the following Turing machine. On input w:

1. If w does not encode a Turing machine, empty the tape and accept
2. Otherwise, replace every occurrence of r in w_δ by t and accept

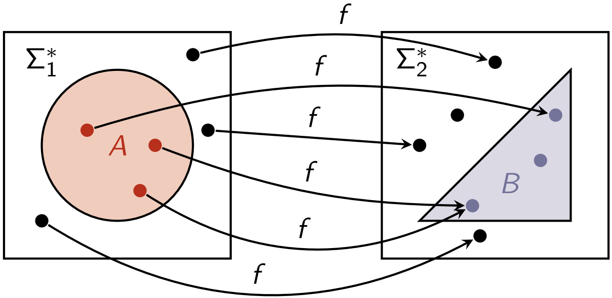

Σ_1^*
A
Σ_2^*
B



Definition

Let $A \subseteq \Sigma_1^*$ and $B \subseteq \Sigma_2^*$ be languages. We say that A is mapping reducible to B, written $A \leq_m B$, if and only if

1. there is a computable function $f : \Sigma_1^* \to \Sigma_2^*$ such that
2. for every $w \in \Sigma_1^*$: $w \in A \Leftrightarrow f(w) \in B$



Definition

Let $A \subseteq \Sigma_1^*$ and $B \subseteq \Sigma_2^*$ be languages. We say that A is mapping reducible to B, written $A \leq_m B$, if and only if

1. there is a computable function $f\colon \Sigma_1^* \to \Sigma_2^*$ such that
2. for every $w \in \Sigma_1^*$: $w \in A \Leftrightarrow f(w) \in B$

Quiz 1

Let $E = \{w \in \{0, 1, \ldots, 9\}^+ \mid w \text{ is even}\}$
and $O = \{w \in \{0, 1, \ldots, 9\}^+ \mid w \text{ is odd}\}$

Show that $E \leq_m O$

Quiz 1

Let $E = \{w \in \{0, 1, \dots, 9\}^+ \mid w \text{ is even}\}$
and $O = \{w \in \{0, 1, \dots, 9\}^+ \mid w \text{ is odd}\}$

Show that $E \leq_m O$

Choose $f(w) = w + 1$ and show that $w \in E \iff f(w) \in O$

Quiz 2

Task 1: Does $A \leq_m B$ imply $B \leq_m A$?

Quiz 2

Task 1: Does $A \leq_m B$ imply $B \leq_m A$?

No (explained next)

Task 2:

Choose a computable problem L and argue that $L \leq_m \mathrm{HP}$

Quiz 2

Task 1: Does $A \leq_m B$ imply $B \leq_m A$?

No (explained next)

Task 2:
Choose a computable problem L and argue that $L \leq_m \mathrm{HP}$

Task 3:
What if you could also show $\mathrm{HP} \leq_m L$?

Quiz 2

Task 1: Does $A \leq_m B$ imply $B \leq_m A$?

No (explained next)

Task 2:
Choose a computable problem L and argue that $L \leq_m \mathrm{HP}$

Task 3:
What if you could also show $\mathrm{HP} \leq_m L$?

Then HP would be computable

Agenda

A Convention to Save Space

- In the following, we may write things like

  "$\ulcorner M \urcorner$ accepts w"

- What we mean is

  "$\ulcorner M \urcorner$ is the encoding of a Turing machine M and M accepts w"

HP $\leq_m$ AP

Theorem

HP $\leq_m$ AP

HP $\leq_m$ AP

$$f(w) = \begin{cases} \ulcorner M' \urcorner & \text{if } w = \ulcorner M \urcorner \text{ and } M' \text{ is obtained from } M \\ & \text{by replacing every occurrence of } r \text{ by } t \\ \varepsilon & \text{if } w \text{ is not an encoding of a Turing machine} \end{cases}$$

$\mathrm{HP} \leq_m \mathrm{AP}$

$$f(w) = \begin{cases} \ulcorner M' \urcorner & \text{if } w = \ulcorner M \urcorner \text{ and } M' \text{ is obtained from } M \\ & \text{by replacing every occurrence of } r \text{ by } t \\ \varepsilon & \text{if } w \text{ is not an encoding of a Turing machine} \end{cases}$$

Proof

- f is computable. Hence, g defined below is also computable:

$$g(w) = \begin{cases} \langle f(\ulcorner M \urcorner), w' \rangle & w = \langle \ulcorner M \urcorner, w' \rangle \text{ such that } w' \text{ is an input for } M \\ \varepsilon & \text{otherwise} \end{cases}$$

- We show $w \in \mathrm{HP} \Leftrightarrow g(w) \in \mathrm{AP}$:

$$f(w) = \begin{cases} \ulcorner M' \urcorner & \text{if } w = \ulcorner M \urcorner \text{ and } M' \text{ is obtained from } M \\ & \text{by replacing every occurrence of } r \text{ by } t \\ \varepsilon & \text{if } w \text{ is not an encoding of a Turing machine} \end{cases}$$

Proof

- f is computable. Hence, g defined below is also computable:

$$g(w) = \begin{cases} \langle f(\ulcorner M \urcorner), w' \rangle & w = \langle \ulcorner M \urcorner, w' \rangle \text{ such that } w' \text{ is an input for } M \\ \varepsilon & \text{otherwise} \end{cases}$$

- We show $w \in \mathrm{HP} \Leftrightarrow g(w) \in \mathrm{AP}$:
  - Let $w \in \mathrm{HP}$

$$f(w) = \begin{cases} \ulcorner M' \urcorner & \text{if } w = \ulcorner M \urcorner \text{ and } M' \text{ is obtained from } M \\ & \text{by replacing every occurrence of } r \text{ by } t \\ \varepsilon & \text{if } w \text{ is not an encoding of a Turing machine} \end{cases}$$

Proof

- f is computable. Hence, g defined below is also computable:

$$g(w) = \begin{cases} \langle f(\ulcorner M \urcorner), w' \rangle & w = \langle \ulcorner M \urcorner, w' \rangle \text{ such that } w' \text{ is an input for } M \\ \varepsilon & \text{otherwise} \end{cases}$$

- We show $w \in \mathrm{HP} \Leftrightarrow g(w) \in \mathrm{AP}$:
  - Let $w \in \mathrm{HP}$
  - Then, $w = \langle \ulcorner M \urcorner, w' \rangle$ such that M halts on w'

$$f(w) = \begin{cases} \ulcorner M' \urcorner & \text{if } w = \ulcorner M \urcorner \text{ and } M' \text{ is obtained from } M \\ & \text{by replacing every occurrence of } r \text{ by } t \\ \varepsilon & \text{if } w \text{ is not an encoding of a Turing machine} \end{cases}$$

Proof

- f is computable. Hence, g defined below is also computable:

$$g(w) = \begin{cases} \langle f(\ulcorner M \urcorner), w' \rangle & w = \langle \ulcorner M \urcorner, w' \rangle \text{ such that } w' \text{ is an input for } M \\ \varepsilon & \text{otherwise} \end{cases}$$

- We show $w \in \mathrm{HP} \Leftrightarrow g(w) \in \mathrm{AP}$:
  - Let $w \in \mathrm{HP}$
  - Then, $w = \langle \ulcorner M \urcorner, w' \rangle$ such that M halts on w'
  - Then, $f(\ulcorner M \urcorner) = \ulcorner M' \urcorner$ accepts w', i.e., $g(w) = \langle f(\ulcorner M \urcorner), w' \rangle \in \mathrm{AP}$

$\mathrm{HP} \leq_m \mathrm{AP}$

$$f(w) = \begin{cases} \ulcorner M' \urcorner & \text{if } w = \ulcorner M \urcorner \text{ and } M' \text{ is obtained from } M \\ & \text{by replacing every occurrence of } r \text{ by } t \\ \varepsilon & \text{if } w \text{ is not an encoding of a Turing machine} \end{cases}$$

Proof

- f is computable. Hence, g defined below is also computable:

$$g(w) = \begin{cases} \langle f(\ulcorner M \urcorner), w' \rangle & w = \langle \ulcorner M \urcorner, w' \rangle \text{ such that } w' \text{ is an input for } M \\ \varepsilon & \text{otherwise} \end{cases}$$

- We show $w \in \mathrm{HP} \Leftrightarrow g(w) \in \mathrm{AP}$:
  - Let $w \notin \mathrm{HP}$

$$f(w) = \begin{cases} \ulcorner M' \urcorner & \text{if } w = \ulcorner M \urcorner \text{ and } M' \text{ is obtained from } M \\ & \text{by replacing every occurrence of } r \text{ by } t \\ \varepsilon & \text{if } w \text{ is not an encoding of a Turing machine} \end{cases}$$

Proof

- f is computable. Hence, g defined below is also computable:

$$g(w) = \begin{cases} \langle f(\ulcorner M \urcorner), w' \rangle & w = \langle \ulcorner M \urcorner, w' \rangle \text{ such that } w' \text{ is an input for } M \\ \varepsilon & \text{otherwise} \end{cases}$$

- We show $w \in \mathrm{HP} \Leftrightarrow g(w) \in \mathrm{AP}$:
  - Let $w \notin \mathrm{HP}$
  1. If $g(w) = \varepsilon$, then $g(w) \notin \mathrm{AP}$

$\mathrm{HP} \leq_m \mathrm{AP}$

$$f(w) = \begin{cases} \ulcorner M' \urcorner & \text{if } w = \ulcorner M \urcorner \text{ and } M' \text{ is obtained from } M \\ & \text{by replacing every occurrence of } r \text{ by } t \\ \varepsilon & \text{if } w \text{ is not an encoding of a Turing machine} \end{cases}$$

Proof

- f is computable. Hence, g defined below is also computable:

$$g(w) = \begin{cases} \langle f(\ulcorner M \urcorner), w' \rangle & w = \langle \ulcorner M \urcorner, w' \rangle \text{ such that } w' \text{ is an input for } M \\ \varepsilon & \text{otherwise} \end{cases}$$

- We show $w \in \mathrm{HP} \Leftrightarrow g(w) \in \mathrm{AP}$:
  - Let $w \notin \mathrm{HP}$
  1. If $g(w) = \varepsilon$, then $g(w) \notin \mathrm{AP}$
  2. Otherwise, $w = \langle \ulcorner M \urcorner, w' \rangle$ s.t. M does not halt on w'

$\mathbf{HP} \le_m \mathbf{AP}$

$$f(w) = \begin{cases} \ulcorner M' \urcorner & \text{if } w = \ulcorner M \urcorner \text{ and } M' \text{ is obtained from } M \\ & \text{by replacing every occurrence of } r \text{ by } t \\ \varepsilon & \text{if } w \text{ is not an encoding of a Turing machine} \end{cases}$$

Proof

- f is computable. Hence, g defined below is also computable:

$$g(w) = \begin{cases} \langle f(\ulcorner M \urcorner), w' \rangle & w = \langle \ulcorner M \urcorner, w' \rangle \text{ such that } w' \text{ is an input for } M \\ \varepsilon & \text{otherwise} \end{cases}$$

- We show $w \in \mathrm{HP} \Leftrightarrow g(w) \in \mathrm{AP}$:
  - Let $w \notin \mathrm{HP}$

  1. If $g(w) = \varepsilon$, then $g(w) \notin \mathrm{AP}$

  2. Otherwise, $w = \langle \ulcorner M \urcorner, w' \rangle$ s.t. M does not halt on w'
    - Then, $f(\ulcorner M \urcorner) = \ulcorner M' \urcorner$ does not halt on w'

$$f(w) = \begin{cases} \ulcorner M' \urcorner & \text{if } w = \ulcorner M \urcorner \text{ and } M' \text{ is obtained from } M \\ & \text{by replacing every occurrence of } r \text{ by } t \\ \varepsilon & \text{if } w \text{ is not an encoding of a Turing machine} \end{cases}$$

Proof

- f is computable. Hence, g defined below is also computable:

$$g(w) = \begin{cases} \langle f(\ulcorner M \urcorner), w' \rangle & w = \langle \ulcorner M \urcorner, w' \rangle \text{ such that } w' \text{ is an input for } M \\ \varepsilon & \text{otherwise} \end{cases}$$

- We show $w \in \mathrm{HP} \Leftrightarrow g(w) \in \mathrm{AP}$:
  - Let $w \notin \mathrm{HP}$
  1. If $g(w) = \varepsilon$, then $g(w) \notin \mathrm{AP}$
  2. Otherwise, $w = \langle \ulcorner M \urcorner, w' \rangle$ s.t. M does not halt on w'
     - Then, $f(\ulcorner M \urcorner) = \ulcorner M' \urcorner$ does not halt on w'
  - In both cases above, $g(w) = \langle f(\ulcorner M \urcorner), w' \rangle \notin \mathrm{AP}$

AP $\leq_m$ HP

$$f(w) = \begin{cases} \langle \ulcorner M' \urcorner, w' \rangle & \text{if } w = \langle \ulcorner M \urcorner, w' \rangle \text{ and } M' \text{ is obtained from } M \text{ by} \\ & \text{replacing every occurrence of } r \text{ by a looping} \\ & \text{state (and adding relevant transitions)} \\ \varepsilon & \text{otherwise} \end{cases}$$

$$f(w) = \begin{cases} \langle \ulcorner M' \urcorner, w' \rangle & \text{if } w = \langle \ulcorner M \urcorner, w' \rangle \text{ and } M' \text{ is obtained from } M \text{ by} \\ & \text{replacing every occurrence of } r \text{ by a looping} \\ & \text{state (and adding relevant transitions)} \\ \varepsilon & \text{otherwise} \end{cases}$$

f is computable. We show $w \in \mathrm{AP} \Leftrightarrow f(w) \in \mathrm{HP}$:

- Let $w \in \mathrm{AP}$
- Then, $w = \langle \ulcorner M \urcorner, w' \rangle$ such that M accepts w'
- Then, M' accepts w' as well, and in particular halts on w'
- Thus, $f(w) = \langle \ulcorner M' \urcorner, w' \rangle \in \mathrm{HP}$

AP $\leq_m$ HP

$$f(w) = \begin{cases} \langle \ulcorner M' \urcorner, w' \rangle & \text{if } w = \langle \ulcorner M \urcorner, w' \rangle \text{ and } M' \text{ is obtained from } M \text{ by} \\ & \text{replacing every occurrence of } r \text{ by a looping} \\ & \text{state (and adding relevant transitions)} \\ \varepsilon & \text{otherwise} \end{cases}$$

f is computable. We show $w \in \mathrm{AP} \Leftrightarrow f(w) \in \mathrm{HP}$:

- Let $w \notin \mathrm{AP}$

1. If $f(w) = \varepsilon$, then $f(w) \notin \mathrm{HP}$
2. Otherwise, $w = \langle \ulcorner M \urcorner, w' \rangle$ such that M does not accept w'
    - Hence, either M rejects w' or M does not halt on w'
    - In both cases, $f(\langle \ulcorner M \urcorner, w' \rangle) = \langle \ulcorner M' \urcorner, w' \rangle$, and M' does not halt on w'

- In both cases above, $f(w) = \langle \ulcorner M' \urcorner, w' \rangle \notin \mathrm{HP}$

Agenda

Main Theorem

Theorem

Let $A \leq_m B$. Then:

- *If B is computable, then A is computable*

Main Theorem

Theorem

Let $A \leq_m B$. Then:

- *If B is computable, then A is computable*
- *If A is not computable, then B is not computable*

Main Theorem

Theorem

Let $A \leq_m B$. Then:

- *If B is computable, then A is computable*
- *If A is not computable, then B is not computable*
- *If B is computably-enumerable, then A is computably-enumerable*

Main Theorem

Theorem

Let $A \leq_m B$. Then:

- *If B is computable, then A is computable*
- *If A is not computable, then B is not computable*
- *If B is computably-enumerable, then A is computably-enumerable*
- *If A is not computably-enumerable, then B is not computably-enumerable*

Main Theorem

Theorem

Let $A \leq_m B$. *Then:*

- *If* B *is computable, then* A *is computable*
- *If* A *is not computable, then* B *is not computable*
- *If* B *is computably-enumerable, then* A *is computably-enumerable*
- *If* A *is not computably-enumerable, then* B *is not computably-enumerable*
- $\overline{A} \leq_m \overline{B}$

Main Theorem

Theorem

Let $A \leq_m B$. Then:

- *If B is computable, then A is computable*
- *If A is not computable, then B is not computable*
- *If B is computably-enumerable, then A is computably-enumerable*
- *If A is not computably-enumerable, then B is not computably-enumerable*
- $\overline{A} \leq_m \overline{B}$

Corollary

AP is not computable, but computably-enumerable

Collatz

Recall Lecture 1: Does the following algorithm return `True` for every possible input $n \geq 1$?

```python
def collatz(n):
  while(n > 1):
    if n%2 == 0:
      n = n/2
    else:
      n = 3*n+1
  return True
```

Collatz

Recall Lecture 1: Does the following algorithm return `True` for every possible input $n \geq 1$?

```python
def collatz(n):
  while(n > 1):
    if n%2 == 0:
      n = n/2
    else:
      n = 3*n+1
  return True
```

Note

If we were able to compute whether a DTM halts on every input, we could use it to solve the Collatz problem. However...

Non-computable Problems

The following problems are all non-computable:

- $\mathrm{HP} = \{\langle \ulcorner M \urcorner, w \rangle \mid M \text{ halts on input } w\}$
- $\mathrm{AP} = \{\langle \ulcorner M \urcorner, w \rangle \mid w \in L(M)\}$
- $\mathrm{NEP} = \{\ulcorner M \urcorner \mid L(M) \neq \emptyset\}$
- $\mathrm{UHP} = \{\ulcorner M \urcorner \mid M \text{ halts on every input}\}$
- $\mathrm{EQP} = \{\langle \ulcorner M \urcorner, \ulcorner M' \urcorner \rangle \mid L(M) = L(M')\}$
- The Entscheidungsproblem

There are many more in logics, automata theory, math, etc.

Non-computable Problems

The following problems are all non-computable:

- $\mathrm{HP} = \{\langle \ulcorner M \urcorner, w\rangle \mid M \text{ halts on input } w\}$
- $\mathrm{AP} = \{\langle \ulcorner M \urcorner, w\rangle \mid w \in L(M)\}$
- $\mathrm{NEP} = \{\ulcorner M \urcorner \mid L(M) \neq \emptyset\}$
- $\mathrm{UHP} = \{\ulcorner M \urcorner \mid M \text{ halts on every input}\}$
- $\mathrm{EQP} = \{\langle \ulcorner M \urcorner, \ulcorner M' \urcorner\rangle \mid L(M) = L(M')\}$
- The Entscheidungsproblem

There are many more in logics, automata theory, math, etc.

- Given six 3×3 integer matrices, can they be multiplied in some order (with possible repetitions) to yield the zero matrix?

Rice's Theorem

In a very specific sense, every interesting question about Turing machines is not computable!

Definition

Let $L \subseteq \Sigma^*$

- L is nontrivial if $L \neq \emptyset$ and $L \neq \Sigma^*$
- L is semantic if $\ulcorner M \urcorner \in L$ and $L(M) = L(M')$ implies $\ulcorner M' \urcorner \in L$ (membership of Turing-machine encodings in L only depends on the accepted language)

Rice's Theorem

In a very specific sense, every interesting question about Turing machines is not computable!

Definition

Let $L \subseteq \Sigma^*$

- L is nontrivial if $L \neq \emptyset$ and $L \neq \Sigma^*$
- L is semantic if $\ulcorner M \urcorner \in L$ and $L(M) = L(M')$ implies $\ulcorner M' \urcorner \in L$ (membership of Turing-machine encodings in L only depends on the accepted language)

Examples

- $\{\ulcorner M \urcorner \mid M \text{ accepts } 09022024\}$ is nontrivial and semantic
- $\{\ulcorner M \urcorner \mid M \text{ accepts at least ten words } w \text{ with } |w| > 11\}$ is nontrivial and semantic
- $\{\ulcorner M \urcorner \mid M \text{ has at least twenty states}\}$ is nontrivial but not semantic

Rice's Theorem

In a very specific sense, every interesting question about Turing machines is not computable!

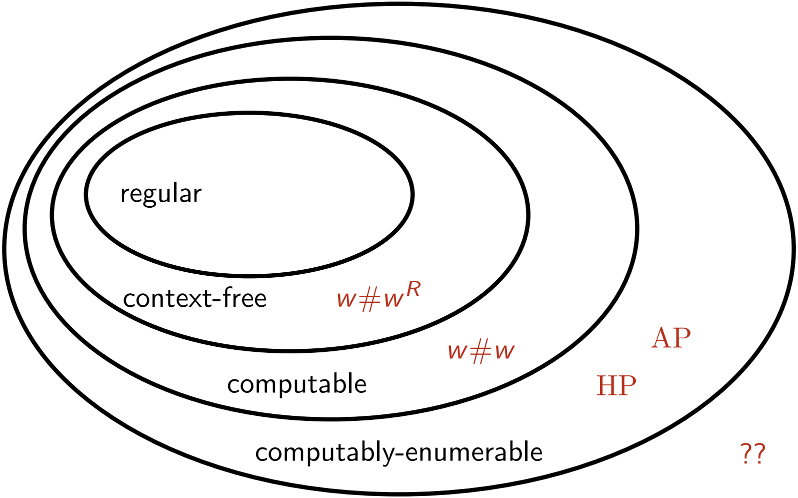
Definition

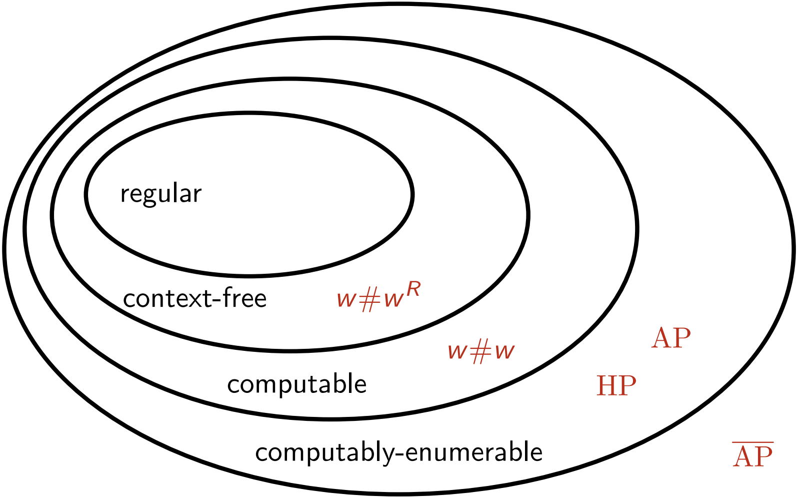
Let $L \subseteq \Sigma^*$

- L is nontrivial if $L \neq \emptyset$ and $L \neq \Sigma^*$
- L is semantic if $\ulcorner M \urcorner \in L$ and $L(M) = L(M')$ implies $\ulcorner M' \urcorner \in L$ (membership of Turing-machine encodings in L only depends on the accepted language)

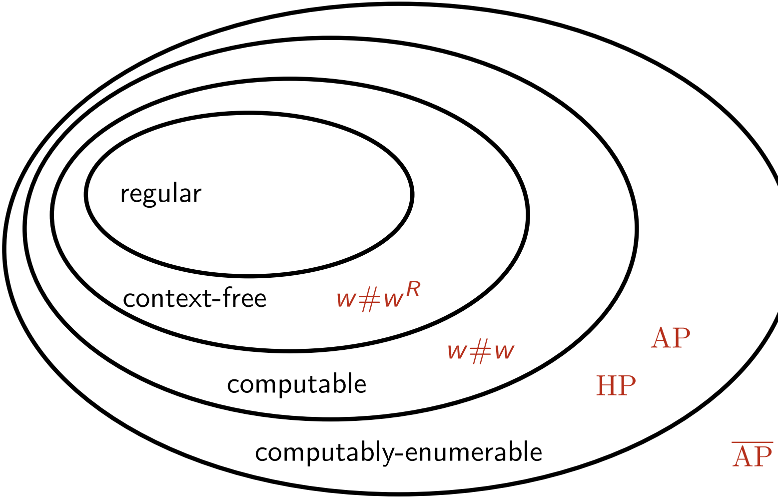
Theorem (Rice 1951)

Every nontrivial semantic language is non-computable


regular
context-free
$w\#w^R$
$w\#w$
computable
AP
HP
computably-enumerable
??


regular
context-free
$w\#w^R$
$w\#w$
computable
AP
HP
computably-enumerable
$\overline{\mathrm{AP}}$

Exercise



Note: AP is computably-enumerable. Thus, computably-enumerable languages are not closed under complementation (cf. Lecture 9)

Conclusion

We have seen

- (Mapping) Reductions,
- More non-computable problems and how to prove them non-computable via reductions, and
- Rice's theorem: everything "interesting" about Turing machines is not computable
- Consequence: everything "interesting" about programs is not computable

Reading

In "Computability and Complexity":

- Section 2.8 on reductions