

CS244: THEORY OF COMPUTATION

Fu Song
ShanghaiTech University

Fall 2022

Outline

Advanced Topics in Computability Theory

The Recursion Theorem

Decidability of Logical Theories

Turing Reducibility

Outline

Advanced Topics in Computability Theory

The Recursion Theorem

Decidability of Logical Theories

Turing Reducibility

Can a machine reproduce itself?

Self-Reference

Lemma

There is a computable function $f : \Sigma^ \rightarrow \Sigma^*$, where if w is any string, $f(w)$ is the description of a Turing machine P_w that prints out w and then halts.*

P_f on input w : (computable function $f : \Sigma^* \rightarrow \Sigma^*$)

1. Construct a TM P_w :

P_w on input x :

1. Erase input.
2. Write w on the tape.
3. Halt.

2. Output $\langle P_w \rangle$.

Self-Reproduce TM

A TM SELF such that $\langle \text{SELF} \rangle = \langle AB \rangle$

- ▶ A produces $\langle B \rangle$ and passes control to B
- ▶ B produces $\langle A \rangle$
- ▶ AB produces $\langle AB \rangle$

How to implement A and B ?

$$A = P_{\langle B \rangle} \text{ and } B = P_{\langle A \rangle}?$$

Self-Reproduce TM

- ▶ $A = P_{\langle B \rangle}$, A writes $\langle B \rangle$ into the tape
- ▶ B constructs A based on the output $\langle B \rangle$ of A

B on input $\langle M \rangle$:

1. Compute $f(\langle M \rangle)$ that is $P_{\langle M \rangle}$. ($P_{\langle B \rangle} = A$ when $M = B$)
2. Combine $P_{\langle M \rangle}$ with $\langle M \rangle$ to make a complete TM SELF.
3. Output $\langle \text{SELF} \rangle$

Behavior of SELF

- ▶ $A = P_{\langle B \rangle}$, A writes $\langle B \rangle$ into the tape
- ▶ B constructs A based on the output $\langle B \rangle$ of A

B on input $\langle M \rangle$:

1. Compute $q(\langle M \rangle)$ that is $P_{\langle M \rangle}$. ($P_{\langle B \rangle} = A$ when $M = B$)
2. Combine $P_{\langle M \rangle}$ with $\langle M \rangle$ to make a complete TM SELF.
3. Output $\langle \text{SELF} \rangle$

Behavior of SELF

1. First A runs. It prints $\langle B \rangle$ on the tape.
2. B starts. It looks at the tape and finds its input, $\langle B \rangle$.
3. B calculates $f(\langle B \rangle)$ that is $\langle P_{\langle B \rangle} \rangle = \langle A \rangle$.
4. B combines that with $\langle A \rangle$ and $\langle B \rangle$ into the TM SELF.
5. B prints $\langle \text{SELF} \rangle$ and halts.

Recursion Theorem

Similar to SELF,
a TM M has ability to refer to its own description $\langle M \rangle$

This idea can be generalized into **recursion theorem** which allows a TM M to obtain its own description $\langle M \rangle$ and perform computation with $\langle M \rangle$ instead of just printing $\langle M \rangle$

Theorem (Recursion theorem)

Let T be a Turing machine that computes a function $t : \Sigma^ \times \Sigma^* \rightarrow \Sigma^*$.
There is a Turing machine R that computes a function $r : \Sigma^* \rightarrow \Sigma^*$,
where for every w ,*

$$r(w) = t(\langle R \rangle, w)$$

Recursion Theorem

Theorem (Recursion theorem)

Let T be a Turing machine that computes a function $t : \Sigma^* \times \Sigma^* \rightarrow \Sigma^*$.
There is a Turing machine R that computes a function $r : \Sigma^* \rightarrow \Sigma^*$,
where for every w ,

$$r(w) = t(\langle R \rangle, w)$$

R on input w :

- ▶ $A = P_{\langle BT \rangle}$, A writes $\langle BT \rangle$ into the tape following w
- ▶ B constructs A based on the output $\langle BT \rangle$ of A

B on input $\langle M \rangle$:

1. Compute $f(\langle M \rangle)$ that is $P_{\langle M \rangle}$. ($P_{\langle BT \rangle} = A$ when $M = BT$)
2. Combine $P_{\langle M \rangle}$ with $w\langle BT \rangle$ and write $\langle R, w \rangle$ into the tap
3. Pass control to T

Applications of Recursion Theorem

Theorem

A_{TM} is *not* decidable.

Recall that we prove this via the diagonalization method. We can prove this via the recursion theorem

B on input w :

1. Obtain via the recursion theorem, $\langle B \rangle$.
2. Run the decider R of A_{TM} on $\langle B, w \rangle$.
3. If R *accepts*, then *reject*. If R *rejects*, then *accept*.

Applications of Recursion Theorem

$\text{MIN}_{\text{TM}} = \{\langle M \rangle \mid M \text{ is a minimal TM that is equivalent to } M\}.$

Theorem

MIN_{TM} is not Turing-recognizable.

C on input w :

1. Obtain via the recursion theorem, $\langle C \rangle$.
 2. Run the enumerator E of MIN_{TM} until a TM D appears with $|\langle D \rangle| > |\langle C \rangle|$.
 3. Simulate D on w .
- ▶ D and C are equivalent
 - ▶ $|\langle D \rangle| > |\langle C \rangle|$
 - ▶ Then, $\langle D \rangle \notin \text{MIN}_{\text{TM}}$, contradiction with Item 2

Applications of Recursion Theorem

Theorem

Let $f : \Sigma^ \rightarrow \Sigma^*$ be a computable function. There exists a TM M such that $f(\langle M \rangle)$ describes a TM equivalent to M .*

M on input w :

1. Obtain via the recursion theorem, $\langle M \rangle$.
2. Compute $f(\langle M \rangle)$ to obtain a description of a TM M' .
3. Simulate M' on w .

Then, M and M' are equivalent.

Outline

Advanced Topics in Computability Theory

The Recursion Theorem

Decidability of Logical Theories

Turing Reducibility

Logical Theories

Definition (Model)

A **model** M is a tuple (U, P_1, \dots, P_k) , where

- ▶ U is the **universe**
- ▶ $P_i : X^r \rightarrow \{\text{Ture}, \text{False}\}$ for $1 \leq i \leq k$ is a r -arity **relation** for some $r \in \mathbb{N}$

Definition (Logical formulae)

Formulae over M are defined by the following syntax:

$$\phi ::= P_i(x_1, \dots, x_r) \mid \phi_1 \wedge \phi_2 \mid \phi_1 \vee \phi_2 \mid (\phi) \mid \neg \phi \mid \exists x. [\phi] \mid \forall x. [\phi]$$

where x, x_1, \dots, x_r are variables over U and P_i is a r -arity relation in M .

Logical Theories

Definition (Sentence)

A variable x in a formula ϕ is called **free variable** if it is not bound within the scope of a quantifier.

A formula ϕ is called **sentence** if it does not have any free variables.

Definition (PNF)

A formula ϕ is in **prenex normal form** if

$$\phi = Q_1 x_1 . Q_2 x_2 . \cdots . Q_k x_k . [\psi]$$

where $Q_1, \cdots Q_k \in \{\forall, \exists\}$ and ψ is a formula without quantifiers.

Language of Logical Theories

$$L(M) = \{\phi \mid \phi \text{ is true in the model } M\}$$

- ▶ $\forall x. \forall y. [x \leq y \vee y \leq x]$ is true in $M = (\mathbb{N}, \leq)$
- ▶ $\forall x. \forall y. [x < y \vee y < x]$ is false in $M = (\mathbb{N}, <)$

Language of Logical Theories

Let P be a 3-arity relation such that $P(x_1, x_2, x_3) \equiv x_1 + x_2 = x_3$

$$L(N_+) = \{\phi \mid \phi \text{ is true in the model } N_+ = (\mathbb{N}, P)\}$$

E.g., $\forall x. \exists y. [x + x = y]$ is **true** in N_+ ,
but, $\forall x. \exists y. [y + y = x]$ is **false** in N_+ .

Theorem

$L(N_+)$ is decidable.

We construct an NFA N such that:

$$\phi \text{ is true} \iff \varepsilon \in L(N)$$

Proof (1)

Theorem

$L(N_+)$ is decidable.

- ▶ Suppose $\phi = Q_1x_1.Q_2x_2.\cdots.Q_kx_k.[\psi]$.
- ▶ Let $\phi_i = Q_{i+1}x_{i+1}.Q_{i+2}x_{i+2}.\cdots.Q_kx_k.[\psi]$ for $0 \leq i \leq k$, where $\phi_k = \psi$.
- ▶ Then ϕ_i contains free variables x_1, x_2, \dots, x_i .
- ▶ We show how to construct a NFA N_k from ϕ_k , i.e., ψ .
- ▶ Then, we show how to construct an NFA N_{k-1} from N_k .
- ▶ Finally, N_0 accepts a word iff ϕ is true.

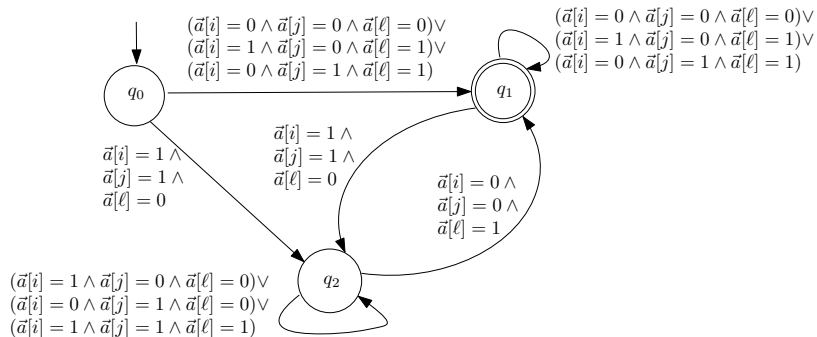
Proof (2)

$$\text{Let } \Sigma_i = \left\{ \begin{bmatrix} 0 \\ \cdot \\ \cdot \\ \cdot \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ \cdot \\ \cdot \\ \cdot \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ \cdot \\ \cdot \\ \cdot \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ \cdot \\ \cdot \\ \cdot \\ 1 \\ 1 \end{bmatrix}, \dots, \begin{bmatrix} 1 \\ \cdot \\ \cdot \\ \cdot \\ 1 \\ 1 \end{bmatrix} \right\}$$

- ▶ Each column is size i and $\Sigma_0 = \{[]\}$
- ▶ A sequence $\vec{a}_1 \vec{a}_2 \cdots \vec{a}_m$ of symbols from Σ_i denotes a valuation of variables x_1, x_2, \dots, x_i
- ▶ The value of x_j is the **reverse** of the binary sequence $\vec{a}_1[j] \vec{a}_2[j] \cdots \vec{a}_m[j]$ of the **j -th row** in $\vec{a}_1 \vec{a}_2 \cdots \vec{a}_m$

Proof (3)

Consider $P(x_i, x_j, x_\ell)$: Construct $N_{P(x_i, x_j, x_\ell)}$



Proof (4)

- ▶ For $\neg P(x_i, x_j, x_\ell)$: construct $\overline{N}_{P(x_i, x_j, x_\ell)}$
- ▶ For $P(x_i, x_j, x_\ell) \wedge P'(x_{i'}, x_{j'}, x_{\ell'})$: construct $N_{P(x_i, x_j, x_\ell)} \cap N_{P'(x_{i'}, x_{j'}, x_{\ell'})}$
- ▶ For $P(x_i, x_j, x_\ell) \vee P'(x_{i'}, x_{j'}, x_{\ell'})$: construct $N_{P(x_i, x_j, x_\ell)} \cup N_{P'(x_{i'}, x_{j'}, x_{\ell'})}$
- ▶ Finally, we get the NFA N_k

Lemma

N_k accepts a word w (that is an assignment of variables x_1, x_2, \dots, x_k)
iff ψ is true under w .

Proof (5)

- ▶ Suppose the NFA N_i for ϕ_i
- ▶ Construct a NFA N_{i-1} for

$$\phi_{i-1} = \exists x_i. Q_{i+1}x_{i+1}. Q_{i+2}x_{i+2}. \cdots Q_k x_k. [\psi]$$

- ▶ N_{i-1} is same as N_i , except that

$$\text{If } q \begin{bmatrix} b_1 \\ \vdots \\ b_{i-1} \\ \textcolor{red}{b_i} \end{bmatrix} \longrightarrow q' \text{ in } N_i, \text{ then, } N_{i-1} \text{ has } q \begin{bmatrix} b_1 \\ \vdots \\ b_{i-1} \end{bmatrix} \longrightarrow q'$$

Lemma

N_{i-1} accepts a word w (that is an assignment of variables x_1, x_2, \dots, x_{i-1}) iff ψ_{i-1} is true under w .

Proof (6)

- ▶ Suppose the NFA N_i for ϕ_i
- ▶ Construct an NFA N_{i-1} for

$$\phi_{i-1} = \forall x_i. Q_{i+1}x_{i+1}. Q_{i+2}x_{i+2}. \cdots Q_k x_k. [\psi]$$

- ▶ Construct an NFA \overline{N}_{i-1} for

$$\neg \phi_{i-1} = \exists x_i. \overline{Q}_{i+1}x_{i+1}. \overline{Q}_{i+2}x_{i+2}. \cdots \overline{Q}_k x_k. [\neg \psi]$$

- ▶ N_{i-1} is the complement of \overline{N}_{i-1}

Lemma

N_{i-1} accepts a word w (that is an assignment of variables x_1, x_2, \dots, x_{i-1}) iff ψ_{i-1} is true under w .

Presburger Arithmetic

A more general decidable theory: Presburger Arithmetic

- ▶ universe: integer \mathbb{Z}
- ▶ atomic formula: $\sum_{i=1}^n a_i x_i \bowtie c$,
 - ▶ a_i 's and c are integer constants
 - ▶ x_i 's are integer variables
 - ▶ $\bowtie \in \{=, \neq, <, >, \leq, \geq, \equiv_m\}$
- ▶ Boolean connectors: \wedge, \vee, \neg
- ▶ Quantifiers: \forall, \exists

Its complexity lies between 2-NEXPTIME and 3-EXPTIME/2-EXPSpace.¹

¹Antoine Durand-Gasselin, Peter Habermehl: On the Use of Non-deterministic Automata for Presburger Arithmetic. CONCUR 2010: 373-387.

An Undecidable Theory: Peano arithmetic

Let P_1 be a 3-arity relation such that $P_1(x_1, x_2, x_3) \equiv x_1 + x_2 = x_3$

Let P_2 be a 3-arity relation such that $P_2(x_1, x_2, x_3) \equiv x_1 \times x_2 = x_3$

$$L(N_{+, \times}) = \{\phi \mid \phi \text{ is true in the model } N_{+, \times} = (\mathbb{N}, P_1, P_2)\}$$

Theorem

$L(N_{+, \times})$ is undecidable.

Proof idea: A_{TM} can be reduced to $L(N_{+, \times})$

1. For a TM M and an input w , construct a formula $\psi_{M,w}$ encoding an **accepting computation history** of M on w
2. $\psi_{M,w}$ contains a free variable x
3. $\exists x. \psi_{M,w}$ is true iff M accepts w

Gödel's Incompleteness Theorem

In any reasonable system of formalizing the notion of provability in number theory, some true statements are unprovable

Definition (Formal Proof)

A **formal proof** of a statement ϕ is a sequence of statements, S_1, S_2, \dots, S_n such that

- ▶ For every $1 \leq i \leq n$, S_i follows from the preceding statements and certain basic axioms about numbers, using simple and precise rules of implication
- ▶ $S_n = \phi$

Gödel's Incompleteness Theorem

The following two reasonable properties of proofs hold:

- ▶ The correctness of a proof of a statement can be checked by machine. Formally, $L_{provable} = \{\langle \phi, \pi \rangle \mid \pi \text{ is a proof of } \phi\}$ is **decidable**.
- ▶ The system of proofs is **sound**. That is, if a statement is provable (i.e., has a proof), it is true.

Theorem

$L(N_{+, \times})$ is Turing-recognizable.

Proof

Theorem

$L(N_{+, \times})$ is Turing-recognizable.

$P_{+, \times}^{TR}$ on input ϕ :

1. For each π possible proof of length $1, 2, \dots$.
2. Run the proof checker R of $L_{provable}$ on $\langle \phi, \pi \rangle$.
3. If R accepts, then **accept**. If R rejects, then **continue Item 2**.

Theorem

*Some true statements in $L(N_{+, \times})$ is **not** provable.*

Assume that all true statements in $L(N_{+, \times})$ are provable. We reduce from $L(N_{+, \times})$.

$P_{+, \times}^{NP}$ on input ϕ :

1. Run the prover of $L(N_{+, \times})$ on ϕ and $\neg\phi$ in parallel.
2. If ϕ is true, then **accept**.
3. If $\neg\phi$ is true, then **reject**.

Then $L(N_{+, \times})$ will be decidable, contradicting the fact that $L(N_{+, \times})$ is undecidable.

Outline

Advanced Topics in Computability Theory

The Recursion Theorem

Decidability of Logical Theories

Turing Reducibility

Turing reducibility

Definition

Language A is **mapping reducible** to language B , written $A \leq_m B$, if there is a computable function $f : \Sigma^* \rightarrow \Sigma^*$, where for every $w \in \Sigma^*$

$$w \in A \iff f(w) \in B.$$

A_{TM} and \overline{A}_{TM} are reducible to one another because a solution to either could be used to solve the other by simply reversing the answer

Definition

An **oracle** for a language B is an external device that is capable of reporting whether any string $w \in B$. An **oracle Turing machine** is a modified Turing machine that has the additional capability of querying an **oracle**. We write M^B to describe an oracle Turing machine that has an oracle for language B .

Turing reducibility

An oracle Turing machine can decide more languages than an ordinary Turing machine can

$T^{A_{TM}}$ on input $\langle M \rangle$:

1. Construct the following TM N :
 N on any input:
 - 1.1 Run M in parallel on all strings in Σ^*
 - 1.2 If M accepts any of these strings, **accept**
2. Query the oracle to determine whether $\langle N, 0 \rangle \in L(A_{TM})$
3. If the oracle answers **NO**, **accept** ; if **YES**, **reject**
 - ▶ If $L(M) \neq \emptyset$, then $L(N) = \Sigma^*$, then $T^{A_{TM}}$ **rejects**
 - ▶ If $L(M) = \emptyset$, then $L(N) = \emptyset$, then $T^{A_{TM}}$ **accepts**

$T^{A_{TM}}$ is a decider of E_{TM} .

Turing reducibility

Definition

Language A is **Turing reducible** to language B , written $A \leq_T B$, if A is **decidable relative to B** , i.e., A uses an oracle of the language B

Theorem

If $A \leq_T B$ and B is decidable, then A is decidable.

Replace the oracle of the language B in oracle TM T^B by the decider of B yields a standard decidable TM for A

Turing reducibility

Oracle TM solve many problems that are not solvable by ordinary Turing machines. But even such a powerful machine cannot decide all languages

Theorem

Let $A'_{\text{TM}} = \{\langle M^{A_{\text{TM}}}, w \rangle \mid M^{A_{\text{TM}}} \text{ is an oracle TM and accepts } w\}$. A'_{TM} is *undecidable* relative to A_{TM} .

Turing reducibility

Theorem

Let $A'_{\text{TM}} = \{\langle M^{\text{ATM}}, w \rangle \mid M^{\text{ATM}} \text{ is an oracle TM and accepts } w\}$. A'_{TM} is *undecidable* relative to A_{TM} .

Assume A'_{TM} is *decidable* relative to A_{TM} , let T^{ATM} be the decider of A'_{TM} .

D^{ATM} on input $\langle M \rangle$:

1. Simulate the decider T^{ATM} of A'_{TM} on $\langle M, \langle M \rangle \rangle$
 2. If T^{ATM} *accepts*, then *reject*
 3. If T^{ATM} *rejects*, then *accept*
- ▶ D^{ATM} accepts $\langle M \rangle$ iff T^{ATM} rejects $\langle M, \langle M \rangle \rangle$ iff M rejects $\langle M \rangle$
 - ▶ Let M be D^{ATM}
 - ▶ D^{ATM} *accepts* $\langle D^{\text{ATM}} \rangle$ iff D^{ATM} *rejects* $\langle D^{\text{ATM}} \rangle$