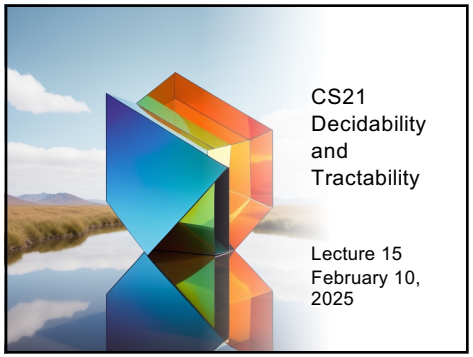


CS21 Decidability and Tractability

Lecture 15
February 10, 2025

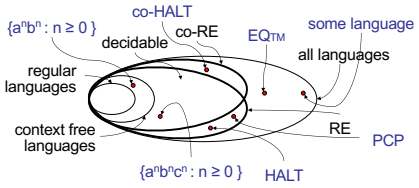
1

Outline

- Recursion Theorem
- Gödel Incompleteness Theorem

2

Summary



3

The Recursion Theorem

- A very useful, and non-obvious, capability of Turing Machines:
  - in the course of computation, can print out a description of itself!
- how is this possible?
  - an example of a program that prints out self:

Print two copies of the following, the 2nd one in quotes:
"Print two copies of the following, the 2nd one in quotes:"

4

The Recursion Theorem

- Why is this useful?
- Example: slick proof that A_{TM} undecidable
  - assume TM M decides A_{TM}
  - construct machine M' as follows:

on input x,
- obtain own description <M'>
- run M on input <M', x>
- if M rejects, accept; if M accepts, reject.

if M' on input x:
- accepts, then M rejects <M', x>, but then M' does not accept!
- rejects, then M accepts <M', x>, but then M' accepts!

5

The Recursion Theorem

- Lemma: there is a computable function

$$q:\Sigma^* \rightarrow \Sigma^*$$

such that q(w) is a description of a TM P_w that prints out w and then halts.

- Proof:
  - on input w, construct TM P_w that has w hard-coded into it; output $<P_w>$

6

The Recursion Theorem

- Warm-up: produce a TM SELF that prints out its own description.
- Two parts:
  - Part A:
    - output a description of B
    - pass control to B.
  - Part B:
    - prepend a description of A
    - done

7

The Recursion Theorem

- Part A:
  - output a description of B
  - pass control to B.
- Part B:
  - prepend a description of A
  - done

Recall: q(w) is a description of a TM P_w that prints out w and then halts.

Note: <A> = q(<B>)

A
- output <B>

B
- read contents of tape
- apply q to it
- prepend* result to tape

*combine with description on tape to produce a complete TM

8

The Recursion Theorem

Note: <A> = q(<B>)

A
- output <B>

B
- read contents of tape
- apply q to it
- prepend result to tape

Recall: q(w) is a description of a TM P_w that prints out w and then halts.

- watch closely as TM AB runs:
- A runs. Tape contents: <B>
- B runs. Tape contents: q(<B>)<B> = <AB>
- AB is our desired machine SELF.

9

The Recursion Theorem

- Lemma: there is a computable function

  $q:\Sigma^* \rightarrow \Sigma^*$

  such that q(w) is a description of a TM P_w that prints out w and then halts.

- Proof:
  - on input w, construct TM P_w that has w hard-coded into it; output <P_w>

10

The Recursion Theorem

- Warm-up: produce a TM SELF that prints out its own description.
- Two parts:
  - Part A:
    - output a description of B
    - pass control to B.
  - Part B:
    - prepend a description of A
    - done

11

The Recursion Theorem

- Part A:
  - output a description of B
  - pass control to B.
- Part B:
  - prepend a description of A
  - done

Recall: q(w) is a description of a TM P_w that prints out w and then halts.

Note: <A> = q(<B>)

A
- output <B>

B
- read contents of tape
- apply q to it
- prepend* result to tape

*combine with description on tape to produce a complete TM

12

The Recursion Theorem

Note: <A> = q(<B>)

A
• output <B>

B
• read contents of tape • apply q to it • prepend result to tape

Recall: q(w) is a description of a TM P_w that prints out w and then halts.

– watch closely as TM AB runs:
– A runs. Tape contents: <B>
– B runs. Tape contents: q(<B>)<B> = <AB>
– AB is our desired machine SELF.

13

The Recursion Theorem

Theorem: Let T be a TM that computes fn:

$t: \Sigma^* \times \Sigma^* \rightarrow \Sigma^*$

There is a TM R that computes the fn:

$r: \Sigma^* \rightarrow \Sigma^*$

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

- This allows "obtain own description" as valid step in TM program
  - first modify TM so that it takes an additional input (that is own description); use at will

14

The Recursion Theorem

Theorem: Let T be a TM that computes fn:

$t: \Sigma^* \times \Sigma^* \rightarrow \Sigma^*$

There is a TM R that computes the fn:

$r: \Sigma^* \rightarrow \Sigma^*$

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

Proof outline: TM R has 3 parts
Part A: output description of BT
Part B: prepend description of A
Part "T": run TM T

15

The Recursion Theorem

Proof details: TM R has 3 parts
Part A: output description of BT
- <A> = q(<BT>)

Part B: prepend description of A
- read contents of tape <BT>
- apply q to it q(<BT>) = <A>
- prepend to tape <ABT>

Part "T": run TM T
- 2nd argument on tape is description of R

16

Gödel Incompleteness Theorem

17

Background

- Hilbert's program (1920's):
  – formalize mathematics in axiomatic form
  – derive all true statements "mechanically" from initial axioms
  – would put mathematicians out of business!
  – very influential proposal
- to start: try for all true statements about the natural numbers ("number theory")

18

Background:

- Kurt Gödel (1931): it is not possible!
- no formalization of number theory can prove all true statements
- stunning result
- considered one of greatest 20th century achievements in mathematics

19

Background

- We will prove using:
  - RE languages and non-RE languages
  - reductions
- Idea:
  - set of all theorems is RE
  - set of all true statements is not RE
- This kind of proof of Gödel's result attributed to Turing (1937).

20

Number Theory

- formal language to express properties of
  $\mathbf{N} = \{0, 1, 2, 3, \ldots\}$
- allowable symbols: parentheses, and
  - variables x,y,z,… ranging over **N**
  - operators + (addition) and * (multiplication)
  - constants 0 (additive id) and 1 (mult. identity)
  - relation = (equality)
  - quantifiers ∀ (for all) and ∃ (exists)
  - propositional operators ∨ (or) ∧ (and) ¬ (not) ⇒ (implies) ⇔ (iff)

21

Number Theory

- can formalize syntax of allowable formulas (skip)
- defining comparison relations:
  - $x \leq y \equiv \exists z \;\; x + z = y$
  - $x < y \equiv \exists z \;\; x + z = y \wedge \neg(z = 0)$

22

Number Theory

- Other natural concepts we will need:
  - quotient q and remainder r when divide x by y
    $\text{INTDIV}(x, y, q, r) \equiv x = qy + r \wedge r < y$
  - y divides x
    $\text{DIV}(y, x) \equiv \exists q \; \text{INTDIV}(x,y,q,0)$
  - x is even
    $\text{EVEN}(x) \equiv \text{DIV}(1+1, x)$
  - x is odd
    $\text{ODD}(x) \equiv \neg\text{EVEN}(x)$

23

Number Theory

- Other natural concepts we will need:
  - x is prime
    $\text{PRIME}(x) \equiv x \geq (1+1) \wedge \forall y \; (\text{DIV}(y, x) \Rightarrow (y = 1 \vee y = x))$
  - x is a power of 2
    $\text{POWER}_2(x) \equiv \forall y \; (\text{DIV}(y, x) \wedge \text{PRIME}(y)) \Rightarrow y = (1+1)$
  - $y = 2^k$ and k^{th} bit of x is 1
    $\text{BIT}(x, y) \equiv \text{POWER}_2(y) \wedge \forall q \; \forall r \; (\text{INTDIV}(x, y, q, r) \Rightarrow \text{ODD}(q))$

24

Number Theory

– $y = 2^k$ and k^{th} bit of x is 1

BIT(x, y) ≡ POWER$_2$(y) ∧ ∀q ∀r (INTDIV(x, y, q, r) ⇒ ODD(q))

y = 10000000000
x = 10101110101110010010 01

q r

25

Number Theory

- A sentence is a formula with no un-quantified variables
  - every number has a successor: ∀x ∃y y = x + 1 (true)
  - every number has a predecessor: ∀x ∃y x = y + 1 (false)
  - not a sentence: x + y = 1
- "number theory" = set of true sentences
  - denoted Th(**N**)

26

Proof systems

- Proof system components:
  - axioms (asserted to be true)
  - rules of inference (mechanical way to derive theorems from axioms)
- axioms for manipulating symbols (e.g.):
  - $(\phi \wedge \psi) \Rightarrow \phi$
  - $(\forall x\, \phi(x)) \Rightarrow \phi(1 + 1 + 1)$
  - $\forall x\, \forall y\, \forall z\, (x = y \wedge y = z \;\Rightarrow\; x = z)$
  - others…

27

Peano Arithmetic

- Peano Arithmetic: proof system for number theory. Axioms:
  - 0 is not a successor
    ∀x ¬(0 = x + 1)
  - the successor function is one-to-one
    ∀x ∀y (x+1 = y+1 ⇒ x = y)
  - 0 is an identity for +
    ∀x x + 0 = x

28

Peano Arithmetic

- + is associative
  ∀x ∀y x + (y + 1) = (x + y) + 1
- multiplying by zero gives 0
  ∀x x*0 = 0
- * distributes over +
  ∀x ∀y x * (y + 1) = (x * y) + x
- induction axiom
  $(\phi(0) \wedge \forall x\, (\phi(x) \Rightarrow \phi(x+1))) \Rightarrow \forall x\, \phi(x)$

29

Peano Arithmetic

- rules of inference:

modus ponens $\dfrac{\phi \qquad \phi \Rightarrow \psi}{\psi}$

generalization $\dfrac{\phi}{\forall x\, \phi}$

30

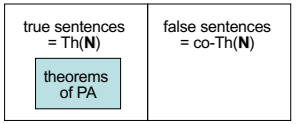
Proof systems

- a proof is a sequence of formulas
  $\phi_1, \phi_2, \phi_3, \ldots \phi_n$
  such that each ϕ_i is either
  - an axiom, or
  - follows from formulas earlier in list from rules of inference
- A sentence is a theorem of the proof system if it has a proof

31

Proof systems

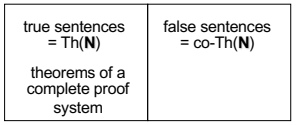
- A proof system is sound if all theorems in that proof system are true (better have this)
- Peano Arithmetic (PA) is sound.



32

Proof systems

- A proof system is complete if all true sentences are theorems in that proof system
- hope to have this (recall Hilbert's program)



33

Incompleteness Theorem

Theorem: Peano Arithmetic is not complete.

(same holds for any reasonable proof system for number theory)

Proof outline:
- the set of theorems of PA is RE
- the set of true sentences (= Th(**N**)) is not RE

34

Incompleteness Theorem

- Lemma: the set of theorems of PA is RE.

- Proof:
  - TM that recognizes the set of theorems of PA:
  - systematically try all possible ways of writing down sequences of formulas
  - accept if encounter a proof of input sentence
    (note: true for any reasonable proof system)

35