Lecture 12

CSE 431
Intro to Theory of Computation

Last time: Accepting computation history of M on input w:

$\#C_0\#C_1\#C_2\#\cdots\#C_t\#$

s.t.

- $C_0 = q_0 w$
- $C_i \vdash_M C_{i+1}$ for $i < t$
- $C_t \in \Gamma^* q_{accept} \Gamma^*$

- exactly one such string or none
- Models that can check this (eg. LBAs) can yield undecidable problems.
  eg. E_{LBA}
- For CFG problems: reversing version
  $\#C_0\#C_1^R\#C_2\#C_3\#\cdots\#C_t\#$

$INTER_{CFG} = \{\langle G_1, G_2\rangle : G_1, G_2 \text{ are CFGs s.t. } L(G_1) \cap L(G_2) \neq \emptyset\}$

$ALL_{CFG} = \{\langle G\rangle : G \text{ is a CFG with } L(G) = \Sigma^*\}$

both undecidable.

Gödel Incompleteness

Logical Theory of Natural Numbers:
$\mathbb{N} = \{0, 1, 2, \ldots\}$
with addition & multiplication

Peano Axioms: symbols $S, +, \cdot, 0$

S = Successor
means "+1"
$S0$ instead of 1
$SS0$ instead of 2
etc

Axioms:

$\forall x (Sx \neq x)$ } Successor
$\forall x \neq 0 \; \exists y (Sy = x)$ }
$\forall x (x + 0 = x)$ } +
$\forall x (x + Sy = S(x + y))$ }
$\forall x (x \cdot 0 = 0)$ } ·
$\forall x (x \cdot Sy = x \cdot y + x)$ }

Infinite number of axioms but easy to enumerate

Induction: For each formula P:
$[\forall x ((P(0) \wedge P(x) \rightarrow P(Sx)) \rightarrow \forall x P(x)$

Methods of Proof:
Predicate Logic inference rules
(as covered in 311)

A proof of Ψ is a sequence of formulas
$\psi_1, \psi_2, \psi_3, \ldots, \psi_t = \Psi$
s.t. every ψ_i is either
- An axiom
- or follows from previous formulas via an inference rule

Ψ is provable iff there is such a proof

Def^n $Th(\mathbb{N}, +, \cdot)$ all statements that are true about $\mathbb{N}, +, \cdot$.

Thm The set of provable statements about $(\mathbb{N}, +, \cdot)$ is Turing-recognizable

Proof Recall: T-rec $\Leftrightarrow$ recursively enumerable r.e.

We build an enumerator for the set of provable statements:

Idea: Create a TM that produces all possible proofs in lexicographic order. At each step can choose either an axiom to include or prior statements to apply an inference rule to.

Output each formula produced

every formula in a proof is a provable statement

dovetail

Note: Though this tries all proofs in lexicographic order, the formulas being proved will not be in order. (Short statements may require very long proofs.)

Every possible proof will be explored so all possible provable statements will eventually be produced

□

Thm $Th(\mathbb{N}, +, \cdot)$ is undecidable

true statements about $\mathbb{N}, +, \cdot$

Proof: Reduction from A_{TM} using accepting computation histories

Idea: Can encode strings as numbers

Accepting computation histories of M on input w $\iff$ natural numbers of a special form

x : candidate natural number of this special form

$A_{TM} \xrightarrow{f} Th(\mathbb{N}, +, \cdot)$

$\langle M, w \rangle \longmapsto \exists x . \phi_{M,w}(x)$

encodes accepting computation of M on input w ← s.t. $\phi_{M,w}(x) = \begin{cases} \text{true if } x \text{ is of the special form} \\ \text{false otherwise} \end{cases}$

Idea: using quantifiers, $+$, $\cdot$ can do things like mod to extract numbers representing individual configurations in the history and check that consecutive configurations satisfy $C_i \vdash_M C_{i+1}$

e.g. "$x < y$" $\exists z (y = x + z)$

"$r = x \bmod y$" $\exists q \exists r ((x = q \cdot y + r) \wedge (r < y))$

i.e. $\exists q \exists r ((x = q \cdot y + r) \wedge (\exists z . y = z + r))$

don't need all the axioms, a finite set Q is enough

Lots of tedious details that we will skip but the function f is computable $A_{TM} \leq_m Th(\mathbb{N}, +, \cdot)$. ∎

Thm ∃ a (true) statement in $Th(\mathbb{N}, +, \cdot)$ that is not provable.

Proof Idea: For each fully quantified statement φ about $(\mathbb{N}, +, \cdot)$ exactly one of φ or $\neg\varphi$ is true

Suppose that every (true) statement in $Th(\mathbb{N}, +, \cdot)$ were provable. (*)

Claim: This would yield a decider for $Th(\mathbb{N}, +, \cdot)$ as follows:

On input φ:
Run enumerator for the set of provable statements in $Th(\mathbb{N}, +, \cdot)$
Either φ or $\neg\varphi$ is true so by assumption (*) one of φ or $\neg\varphi$ will be produced by the enumerator
If φ is produced accept
If $\neg\varphi$ is produced reject.

This would decide $Th(\mathbb{N}, +, \cdot)$ which is impossible so the assumption is false ∎

Gödel found a single explicit statement that is true but not provable. It expresses:

"This statement is not provable."

To produce this, one needs a statement that can talk about itself. Can do this based on:

Recursion Theorem (see 6.1 in Sipser)

Can produce a program that prints out its own code.) Fun exercise

---

Related:

Gödel's Completeness Theorem:

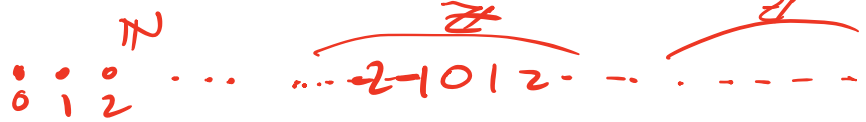
If a set of formulas doesn't yield a provable contradiction then there is a world ("model") where it is true

Consider an enumerable set of axioms A for $Th(\mathbb{N}, +, \cdot)$ and φ s.t. neither φ nor $\neg\varphi$ is provable

$A \cup \{\varphi\}$ doesn't yield a contradiction
$A \cup \{\neg\varphi\}$ doesn't yield a contradiction

true in different worlds

∴ ⇒ there exist "non-standard" models for A



Turing reductions

so far: mapping reductions
$A \leq_m B$ ← one call to B on input $f(x)$, take its answer

Turing: $A \leq_T B$ "A Turing-reducible to B"
iff can use a subroutine for B and call it many times, modifying answers to solve A.

What Turing used to derive TM D from H.
Also for 1st proof for $HALT_{TM}$

Note: $\overline{A}_{TM} \leq_T A_{TM}$ but $\overline{A}_{TM} \not\leq_m A_{TM}$

so $\leq_T$ does not preserve "T-rec"

but $\leq_T$ does preserve "decidable"