CS21
Decidability and Tractability

Lecture 14

February 7, 2018

Outline

- Gödel Incompleteness Theorem

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")

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

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).

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 $\forall$ (for all) and $\exists$ (exists)
  - propositional operators $\wedge$ (and) $\vee$ (or) $\neg$ (not) $\Rightarrow$ (implies) $\Leftrightarrow$ (iff)

Number Theory

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

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

$$\text{ODD}(x) \equiv \neg\, \text{EVEN}(x)$$

Number Theory

- Other natural concepts we will need:
  - x is prime

    $PRIME(x) \equiv x \geq (1+1) \wedge \forall y\ (DIV(y, x) \Rightarrow (y = 1 \vee y = x))$
  - x is a power of 2

    $POWER_2(x) \equiv \forall y\ (DIV(y, x) \wedge PRIME(y)) \Rightarrow y = (1+1)$
  - $y = 2^k$ and k^{th} bit of x is 1

    $BIT(x, y) \equiv POWER_2(y) \wedge \forall q\ \forall r\ (INTDIV(x, y, q, r) \Rightarrow ODD(q))$

Number Theory

– $y = 2^k$ and k^{th} bit of x is 1

$$BIT(x, y) \equiv POWER_2(y) \wedge \forall q \; \forall r \; (INTDIV(x, y, q, r) \Rightarrow ODD(q))$$

y = 10000000000

x = 10101110101 1 1001001001

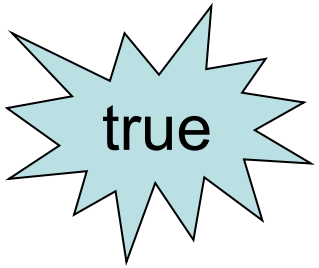
q r

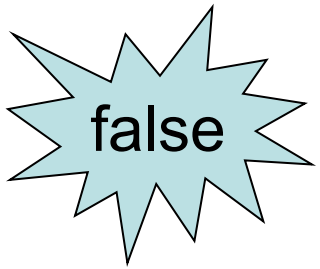
Number Theory

- A sentence is a formula with no un-quantified variables
  - every number has a successor:

    $\forall x \exists y\ y = x + 1$
  - every number has a predecessor:

    $\forall x \exists y\ x = y + 1$
  - not a sentence: $x + y = 1$





- "number theory" = set of true sentences
  - denoted Th(**N**)

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.):
  - $(\varphi \wedge \psi) \Rightarrow \varphi$
  - $(\forall x\ \varphi(x)) \Rightarrow \varphi(1+1+1)$
  - $\forall x\ \forall y\ \forall z\ (x = y \wedge y = z \Rightarrow x = z)$
  - others…

Peano Arithmetic

- Peano Arithmetic: proof system for number theory. Axioms:
  - 0 is not a successor

$$\forall x \neg (0 = x + 1)$$

  - the successor function is one-to-one

$$\forall x \, \forall y \, (x+1 = y+1 \Rightarrow x = y)$$

  - 0 is an identity for +

$$\forall x \; x + 0 = x$$

Peano Arithmetic

– + is associative

$$\forall x \; \forall y \;\; x + (y + 1) = (x + y) + 1$$

– multiplying by zero gives 0

$$\forall x \; x*0 = 0$$

– * distributes over +

$$\forall x \; \forall y \;\; x * (y + 1) = (x * y) + x$$

– induction axiom

$$(\varphi(0) \wedge \forall x \; (\varphi(x) \Rightarrow \varphi(x+1))) \Rightarrow \forall x \; \varphi(x)$$

Peano Arithmetic

- rules of inference:

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

generalization $\dfrac{\varphi}{\forall \mathbf{x}\ \varphi}$

Proof systems

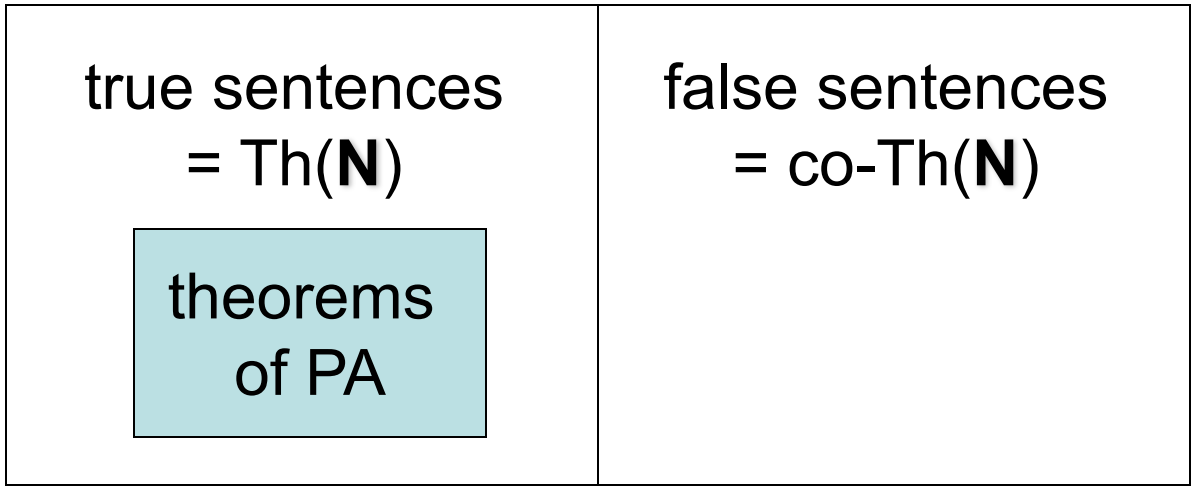
- a proof is a sequence of formulas

  $$\varphi_1, \varphi_2, \varphi_3, \ldots, \varphi_n$$

  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

Proof systems

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



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)

true sentences = Th(**N**) theorems of a complete proof system	false sentences = co-Th(**N**)

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

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)

Incompleteness Theorem

- Lemma: Th(**N**) is not RE

- Proof:
  - reduce from co-HALT (show co-HALT $\leq_m$ Th(**N**))
  - recall co-HALT is not RE

  - what should f(<M, w>) produce?
  - construct γ such that M loops on w $\Leftrightarrow$ γ is true

Incompleteness Theorem

– we will define

$VALCOMP_{M,w}(v) \equiv \ldots$ (details to come)

so that it is true iff v is a (halting) computation history of M on input w

– then define f(<M, w>) to be:

$$\gamma \equiv \neg \exists v \; VALCOMP_{M,w}(v)$$

– YES maps YES?

- <M, w> $\in$ co-HALT $\Rightarrow \gamma$ is true $\Rightarrow \gamma \in$Th(**N**)

– NO maps to NO?

- <M, w> $\notin$ co-HALT $\Rightarrow \gamma$ is false $\Rightarrow \gamma \notin$Th(**N**)

Expressing computation in the language of number theory

- Last time: basic building blocks
  - $x < y \equiv \exists z\ x + z = y \wedge \neg (z = 0)$
  - $\text{INTDIV}(x, y, q, r) \equiv x = qy + r \wedge r < y$
  - $\text{DIV}(y, x) \equiv \exists\ q\ \text{INTDIV}(x,y,q,0)$
  - $\text{PRIME}(x) \equiv$

    $x \geq (1+1) \wedge \forall y\ (\text{DIV}(y, x) \Rightarrow (y = 1 \vee y = x))$

Expressing computation in the language of number theory

– we'll write configurations over an alphabet of size p, where p is a prime that depends on M

– d is a power of p:

$$POWER_p(d) \equiv \forall z\ (DIV(z, d) \wedge PRIME(z)) \Rightarrow z = p$$

– $d = p^k$ and length of v as a p-ary string is k

$$LENGTH(v, d) \equiv POWER_p(d) \wedge v < d$$

Expressing computation in the language of number theory

– the p-ary digit of v at position y is b (assuming y is a power of p):

$$\text{DIGIT}(v, y, b) \equiv$$

$$\exists u\, \exists a\, (v = a + by + upy \wedge a < y \wedge b < p)$$

– the three p-ary digits of v at position y are b,c, and d (assuming y is a power of p):

$$\text{3DIGIT}(v, y, b, c, d) \equiv$$

$$\exists u\, \exists a\, (v = a + by + cpy + dppy + upppy$$

$$\wedge a < y \wedge b < p \wedge c < p \wedge d < p)$$

Expressing computation in the language of number theory

– the three p-ary digits of v at position y "match" the three p-ary digits of v at position z according to M's transition function (assuming y and z are powers of p):

$$\text{MATCH}(v, y, z) \equiv$$

$$\bigvee_{(a,b,c,d,e,f) \in C} \text{3DIGIT}(v, y, a, b, c) \wedge \text{3DIGIT}(v, z, d, e, f)$$

where C = {(a,b,c,d,e,f) : abc in config. C_i can legally change to def in config. C_{i+1}}

Expressing computation in the language of number theory

– all pairs of 3-digit sequences in v up to d that are exactly c apart "match" according to M's transition function (assuming c, d powers of p)

$$\text{MOVE}(v, c, d) \equiv$$
$$\forall y \, (\text{POWER}_p(y) \wedge yppc < d) \Rightarrow \text{MATCH}(v, y, yc)$$

Expressing computation in the language of number theory

– the string v starts with the start configuration of M on input $w = w_1 \ldots w_n$ padded with blanks out to length c (assuming c is a power of p):

$$\text{START}(v, c) \equiv$$

$$\bigwedge_{i = 0,1,2,\ldots, n} \text{DIGIT}(v, p^i, k_i) \wedge p^n < c$$
$$\wedge \forall y \, (\text{POWER}_p(y) \wedge p^n < y < c \Rightarrow \text{DIGIT}(v, y, k))$$

where $k_0k_1k_2k_3 \ldots k_n$ is the p-ary encoding of the start configuration, and k is the p-ary encoding of a blank symbol.

Expressing computation in the language of number theory

– string v has a halt state in it somewhere before position d (assuming d is power of p):

$$HALT(v, d) \equiv$$

$$\exists y \, (POWER_p(y) \wedge y < d \wedge \bigvee_{a \in H} DIGIT(v,y,a))$$

where H is the set of p-ary digits "containing" states q_{accept} or q_{reject}.

Expressing computation in the language of number theory

– string v is a valid (halting) computation history of machine M on string w:

$$\text{VALCOMP}_{M,w}(v) \equiv$$
$$\exists c\ \exists d\ (\text{POWER}_p(c) \wedge c < d \wedge \text{LENGTH}(v, d) \wedge \text{START}(v, c) \wedge \text{MOVE}(v, c, d) \wedge \text{HALT}(v, d))$$

– M does not halt on input w:

$$\neg \exists v\ \text{VALCOMP}_{M,w}(v)$$

Gödel Incompleteness Theorem (an example)

Incompleteness Theorem

v = 036320363205332035320314203124

$$VALCOMP_{M,w}(v) \equiv$$
$$\exists c\, \exists d\, (POWER_p(c) \wedge c < d \wedge LENGTH(v, d) \wedge START(v, c) \wedge MOVE(v, c, d) \wedge HALT(v, d))$$

Incompleteness Theorem

v = 036320363205332035320314203124

d = 10000000000000000000000000000

$VALCOMP_{M,w}(v) \equiv$

$\exists c\ \exists d\ (POWER_p(c) \wedge c < d \wedge LENGTH(v, d) \wedge$
$START(v, c) \wedge MOVE(v, c, d) \wedge HALT(v, d))$

d = p^k and length of v as a p-ary string is k

$LENGTH(v, d) \equiv POWER_p(d) \wedge v < d$

Incompleteness Theorem

$kk_n \ldots k_2 k_1 k_0$

v = 036320363205332035320314203124

c = 100000

p^n=1000

$VALCOMP_{M,w}(v) \equiv$

$\exists c \, \exists d \, (POWER_p(c) \wedge c < d \wedge LENGTH(v, d) \wedge$
$START(v, c) \wedge MOVE(v, c, d) \wedge HALT(v, d))$

v starts with the start configuration of M on input w padded with blanks out to length c:

$START(v, c) \equiv \wedge_{i = 0, \ldots, n} DIGIT(v, p^i, k_i) \wedge p^n < c \wedge$
$\forall y \, (POWER_p(y) \wedge p^n < y < c \Rightarrow DIGIT(v, y, k))$

Incompleteness Theorem

v = 036320363205332035320314203124

yc = 100000
y =1

$VALCOMP_{M,w}(v) \equiv$

$\exists c\ \exists d\ (POWER_p(c) \wedge c < d \wedge LENGTH(v, d) \wedge START(v, c) \wedge MOVE(v, c, d) \wedge HALT(v, d))$

all pairs of 3-digit sequences in v up to d exactly c apart "match" according to M's transition function

$MOVE(v, c, d) \equiv \forall y\ (POWER_p(y) \wedge yppc < d) \Rightarrow MATCH(v, y, yc)$

Incompleteness Theorem

v = 036320363205332035320314203124

yc = 1000000
y =10

$VALCOMP_{M,w}(v) \equiv$

$\exists c\ \exists d\ (POWER_p(c) \wedge c < d \wedge LENGTH(v, d) \wedge START(v, c) \wedge MOVE(v, c, d) \wedge HALT(v, d))$

all pairs of 3-digit sequences in v up to d exactly c apart "match" according to M's transition function

$MOVE(v, c, d) \equiv \forall y\ (POWER_p(y) \wedge yppc < d) \Rightarrow MATCH(v, y, yc)$

Incompleteness Theorem

v = 036320363205332035320314203124

yc = 10000000
y =100

$VALCOMP_{M,w}(v) \equiv$

$\exists c\, \exists d\, (POWER_p(c) \wedge c < d \wedge LENGTH(v, d) \wedge START(v, c) \wedge MOVE(v, c, d) \wedge HALT(v, d))$

all pairs of 3-digit sequences in v up to d exactly c apart "match" according to M's transition function

$MOVE(v, c, d) \equiv \forall y\, (POWER_p(y) \wedge yppc < d) \Rightarrow MATCH(v, y, yc)$

Incompleteness Theorem

halt state

v = 036320363205332035320314203124

y = 10000000000000000000000000000

$VALCOMP_{M,w}(v) \equiv$

$\exists c\ \exists d\ (POWER_p(c) \wedge c < d \wedge LENGTH(v, d) \wedge START(v, c) \wedge MOVE(v, c, d) \wedge HALT(v, d))$

string v has a halt state in it before pos'n d:

$HALT(v, d) \equiv \exists y\ (POWER_p(y) \wedge y < d \wedge \vee_{a \in H}\ DIGIT(v,y,a))$

Incompleteness Theorem

- Lemma: Th(**N**) is not RE

- Proof:
  - reduce from co-HALT (show co-HALT $\leq_m$ Th(**N**))
  - recall co-HALT is not RE

  - constructed γ such that

M loops on w $\Leftrightarrow$ γ is true

Summary

- full-fledged model of computation: TM
- many equivalent models
- Church-Turing Thesis

- encoding of inputs
- Universal TM

Summary

- classes of problems:
  - decidable ("solvable by algorithms")
  - recursively enumerable (RE)
  - co-RE

- counting:
  - not all problems are decidable
  - not all problems are RE

Summary

- diagonalization: HALT is undecidable
- reductions: other problems undecidable
  - many examples
  - Rice's Theorem
- natural problems that are not RE
- Recursion Theorem: non-obvious capability of TMs: printing out own description
- Incompleteness Theorem