Outline

- Gödel Incompleteness Theorem

Number Theory

- formal language to express properties of $\mathbb{N} = \{0, 1, 2, 3, \ldots\}$
- allowable symbols: parentheses, and
  - variables $x, y, z$, … ranging over $\mathbb{N}$
  - operators $+$ (addition) and $\ast$ (multiplication)
  - constants 0 (additive id) and 1 (mult. identity)
  - relation $=$ (equality)
  - quantifiers $\forall$ (for all) and $\exists$ (exists)
  - propositional operators $\lor$ (or), $\land$ (and), $\neg$ (not), $\Rightarrow$ (implies), $\iff$ (iff)

Number Theory

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

Number Theory

- Other natural concepts we will need:
  - quotient $q$ and remainder $r$ when divide $x$ by $y$
    $\text{INTDIV}(x, y, q, r) \equiv x = qy + r \land r < y$
  - $y$ 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
    $\text{PRIME}(x) \equiv x \geq (1 + 1) \land \forall y (\text{DIV}(y, x) \Rightarrow (y = 1 \lor y = x))$
  - $x$ is a power of 2
    $\text{POWER}_2(x) \equiv \forall y (\text{DIV}(y, x) \land \text{PRIME}(y)) \Rightarrow y = (1 + 1)$
  - $y = 2^k$ and $k^{	ext{th}}$ bit of $x$ is 1
    $\text{BIT}(x, y) \equiv \text{POWER}_2(y) \land \forall q \forall r (\text{INTDIV}(x, y, q, r) \Rightarrow \text{ODD}(q))$
**Number Theory**

- $y = 2^k$ and $k$th bit of $x$ is 1

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

$y = 1000000000$

$x = 101011010111001001001$

**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.):
  - $(\phi \land \psi) \Rightarrow \phi$
  - $(\forall x \phi(x)) \Rightarrow \phi(1 + 1 + 1)$
  - $\forall x \forall y \forall z (x = y \land y = z \Rightarrow x = z)$
  - 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 +: $\forall x x + 0 = x$

**Proof systems**

- A sentence is a formula with no unquantified variables
  - every number has a successor: $\forall x \exists y y = x + 1$
  - every number has a predecessor: $\forall x \exists y y = x + 1$
  - not a sentence: $x + y = 1$

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

**Peano Arithmetic**

- rules of inference:
  - modus ponens $\phi \quad \phi \Rightarrow \psi$
  - generalization $\phi \quad \forall x \phi$

- + is associative $\forall x \forall y (x + (y + 1) = (x + y) + 1$
- multiplying by zero gives 0 $\forall x x \cdot 0 = 0$
- * distributes over + $\forall x \forall y x \cdot (y + 1) = (x \cdot y) + x$
- induction axiom $(\phi(0) \land \forall x (\phi(x) \Rightarrow \phi(x+1))) \Rightarrow \forall x \phi(x)$
Proof systems

- a proof is a sequence of formulas $\phi_1, \phi_2, \phi_3, \ldots, \phi_n$ such that each $\phi_i$ 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

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)

- Theorem: Peano Arithmetic is not complete.
  (same holds for any reasonable proof system for number theory)
  - the set of theorems of PA is RE
  - the set of true sentences (= $\text{Th}(\mathbb{N})$) is not RE
  - $\text{Th}(\mathbb{N})$ is not RE
  - co-$\text{Th}(\mathbb{N})$ is not RE
  - what should $f(<M, w>)$ produce?
  - construct $\gamma$ such that $M$ loops on $w \iff \gamma$ 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?
    - \( <M,w> \in \text{co-HALT} \Rightarrow \gamma \text{ is true} \Rightarrow \gamma \in \text{Th} (\mathbb{N}) \)
  - NO maps to NO?
    - \( <M,w> \not\in \text{co-HALT} \Rightarrow \gamma \text{ is false} \Rightarrow \gamma \not\in \text{Th} (\mathbb{N}) \)

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 \( p \):
    \[ \text{POWER}_p(d) \equiv \forall z (\text{DIV}(z, d) \land \text{PRIME}(z)) \Rightarrow z = p \]
  - \( d = p^k \) and length of \( v \) as a \( p \)-ary string is \( k \)
    \[ \text{LENGTH}(v, d) \equiv \text{POWER}_p(d) \land v < d \]

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 \( p \)):
  \[ \text{MATCH}(v, y, z) \equiv \bigvee_{(a,b,c,d,e,f) \in C} \text{DIGIT}(v, a, b, c) \land \text{DIGIT}(v, z, d, e, f) \]
  where \( C = \{(a,b,c,d,e,f) : \text{abc in config, C; can legally change to def in config, C_{i+1}} \} \)

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 \( p \)):

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

Expressing computation in the language of number theory

- Recall: basic building blocks
  - \( x < y \equiv \exists z \ x + z = y \land \neg(z = 0) \)
  - \( \text{INTDIV}(x, y, q, r) \equiv x = qy + r \land r < y \)
  - \( \text{DIV}(y, x) \equiv \exists q \text{ INTDIV}(x,y,q,0) \)
  - \( \text{PRIME}(x) \equiv x \geq (1+1) \land \forall y (\text{DIV}(y, x) \Rightarrow (y = 1 \lor y = x)) \)
Expressing computation in the language of number theory

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

\[
\text{START}(v, c) \equiv \bigwedge_{i=0,1,2,3,...}^{p^n < c} \forall y (\text{POWER}(p^i, y) \land p^n < y < c \Rightarrow \text{DIGIT}(v, y, k))
\]

where \( k_0k_1k_2...k_n \) 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 \( p \)): 

\[
\text{HALT}(v, d) \equiv \exists y (\text{POWER}(y) \land y < d \land \bigwedge_{a \in H} \text{DIGIT}(v, y, a))
\]

where \( H \) is the set of \( p \)-ary digits “containing” states \( q_{\text{accept}} \) or \( q_{\text{reject}} \).

Incompleteness Theorem

• Lemma: \( \text{Th}(N) \) is not RE

• Proof:
  – reduce from co-HALT (show co-HALT \( \leq_m \text{Th}(N) \))
  – recall co-HALT is not RE
  – constructed \( \gamma \) such that
    \[ M \text{ loops on } w \iff \gamma \text{ is true} \]

Incompleteness Theorem

• Lemma: \( \text{Th}(N) \) is not RE

• Proof:
  – reduce from co-HALT (show co-HALT \( \leq_m \text{Th}(N) \))
  – recall co-HALT is not RE
  – constructed \( \gamma \) such that
    \[ M \text{ loops on } w \iff \gamma \text{ is true} \]

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