Recursion Theorem (6.1)

The recursion theorem:

We may assume a TM has access to its own code, so any TM M can use a step of "get your own code $\langle M \rangle$".

Theorem:

A_{TM} is undecidable.

Proof: (using the recursion theorem)

By contradiction, assume TM H decides A_{TM}, then define the following TM B:

On input w:

1. Obtain your own code $\langle B \rangle$.
2. Simulate H on $\langle B, w \rangle$.
    - If H accepts, *reject*.
    - Otherwise, *accept*.

The contradiction: B accepts $w \Leftrightarrow B$ does not accept w, therefore H cannot exist □

Theorem

$MIN_{TM} = \{\langle M \rangle \mid M \; is \; a \; TM \; s.t. \forall N: L(M) = L(N), M \; is \; shorter \; than \; N\}$. MIN_{TM} is not Turing-recognizable.

Proof:

By contradiction, assume MIN_{TM} is Turing-recognizable, therefore there exists a TM E that enumerates MIN_{TM}. Then construct TM C as follows:

On input w:

1. Obtain $\langle C \rangle$.
2. Run E until a TM D appears such that $|\langle D \rangle| > |\langle C \rangle|$.
3. Simulate D on w.

Result: $L(C) = L(D)$ and $\langle C \rangle$ is shorter than $\langle D \rangle$, so E outputting D is an error, since $\langle D \rangle \notin MIN_{TM}$, therefore E does not exist, a contradiction □

The Fixed-Point Theorem (6.8)

Let $t: \Sigma^* \to \Sigma^*$ be a computable function, then there exists a TM F such that $t(\langle F \rangle)$ is also a TM and $L(F) = L\big(t(\langle F \rangle)\big)$ (equivalent to F). The functionality of F is the fixed point, i.e. t does not change the functionality of the input functionality.

Proof:

Define F as follows:

On input w:

1. Obtain $\langle F \rangle$.
2. Apply $G \coloneqq t(\langle F \rangle)$ and simulate G on w.

Clearly F and $t(\langle F \rangle)$ are both equal to G, i.e. they do the same, so F is a fixed point for t □

Note: we consider all strings as TMs, and if they don't encode a proper TM, we address that as a TM that accepts $\emptyset$.

Decidability of Logical Theories (6.2)

Mathematical proofs are simply strings that can be checked by a machine whether they are proper proofs, i.e. correct: the string should simply be a logic derivation derived from a set of axioms.

A-priori it is not clear that everything that is true can be proven. We will see that it depends: for some instances it is the situation, but in others – no. But the fact this is the situation – the incompleteness theorem – can be proven.

Definitions

Consider the following statements:

1. $\forall q \exists p \forall x, y[p > q \wedge (x, y > 1 \Rightarrow x \cdot y \neq p)]$ – This is the theorem of infinity of prime numbers: for any integer q there exists a greater integer p such that p doesn't have any proper divisors (i.e. no two integers $x, y > 1$ exist such that their product equals to p).
2. $\forall a, b, c, n[(a, b, c > 0 \wedge n > 2) \Rightarrow a^n + b^n \neq c^n]$ – This is "Fermat's Conjecture" (proven recently by Andrew Wiles).
3. $\forall q \exists p \forall x, y[p > q \wedge x, y > 1 \Rightarrow x \cdot y \neq p \wedge x \cdot y \neq p + 2]$ – There are infinitely many twin primes (i.e. two consecutive primes distant by 2) – have not been proven.

Quantifiers: $\exists, \forall$ (existential, universal).

Relations: a set of tuples, i.e. $p > q$ is the ">" relation: $\{(p, q) \mid p\ is\ greater\ than\ q\}$ (written in an infix). All pairs in this relation hold $p > q$.

Formulas:

Constructed of atomic formulas (a relation with variables), concatenated by Boolean operators and optional quantifiers (all written at the beginning). A bound variable – a variable that appears in a previous quantifier; a free variable – an unbound variable; for instance:

$\exists x.\, ax > 0 - x$ is a bound variable and a is a free variable.

$\exists x[ax^2 + bx + c = 0 \wedge a \neq 0]$ – this is equivalent ($\Leftrightarrow$) to $b^2 - 4ac \geq 0$ (discriminant) – where the right hand side is a quantifier free form.

Universe and models:

$\varphi = \forall x \forall y[R_1(x, y) \vee R_1(y, x)]$ – this sentence doesn't have a meaning without a model. The model defines from what set do we take x, y from. So over the model $M = (\mathbb{N}, \leq)$ the sentence above is true: $\forall x \in \mathbb{N} \forall y \in \mathbb{N}\ x \leq y \vee y \leq x$.

$(\mathbb{N}, \leq)$ is a model of φ since φ is true in that model. Here, $\mathbb{N}$ is the universe.

We can also say $(\mathbb{Q}, \leq)$ is a model for φ, but $(\mathbb{Q}, <)$ is not – as in the case where $x = y$ φ does not hold true.

$PLUS(x, y, z)$: this is a relation defined to be true if $x + y = z$. For instance:

- $\psi = \forall y \exists x\ PLUS(x, x, y)$ - $(\mathbb{Q}, +)$ is a model for ψ but $(\mathbb{N}, +)$ is not – since for instance for any odd y, there exists no $x \in \mathbb{N}$ such that $x + x = y$.

Theorem (6.12)

$\mathcal{T}(\mathbb{N}, +)$ is decidable, where $\mathcal{T}$ is the theory of a model – the set of all statements that hold true over the given model. So $\mathcal{T}(\mathbb{N}, +)$ is the set of all true statements over the universe $\mathbb{N}$ and the relation $+$.

The theorem states that this set is decidable, i.e. there exists a TM M such that for any φ M decides whether φ is a true statement over the model $(\mathbb{N}, +)$.

Proof:

First, we look at problem 1.32, where we construct a DFA over the alphabet of 3-length binary columns, i.e. $\Sigma = \left\{\begin{bmatrix}0\\0\\0\end{bmatrix}, \begin{bmatrix}0\\0\\1\end{bmatrix}, \dots, \begin{bmatrix}1\\1\\1\end{bmatrix}\right\}$ ($|\Sigma| = 8$). The problem: accepting all strings over Σ where the sum of the first and second rows equals the third row. We construct a DFA with 3 states:

- q_0: signifies a state where we are currently with carry 0.
- q_1: the same, only with carry 1.
- q_{reject}: to get all non-legal transitions (e.g 1+1 with carry 0 go to 1), and loop there forever.

For the proof of the theorem we do something similar over quantifiers. We have a formula $\varphi = Q_1 x_1 \dots Q_l x_l[\psi]$, where Q_i are quantifiers and x_i are variables. The idea is that we can construct for each $i = 1, \dots, l$ a DFA that accepts it. We do that iteratively – we start with ψ alone and for that we use the DFA we constructed for 1.32 (only for perhaps longer tuples, not only 3). Then:

- for $Q_i = \exists$, we just need to check all cases for x_i (or perhaps continue non-deterministically, which is equivalent), and go backwards.
- For $Q_i = \forall$, we simply use the fact $\neg\exists x \neg\psi \equiv \forall\neg\neg\psi \equiv \forall\psi$.