Arithmetical Hierarchy

Klaus Sutner

Carnegie Mellon University

1 The Turing Jump

2 Arithmetical Hierarchy

3 Definability

4 Formal Systems

Recall: Oracles

We can attach an orcale to a Turing machine without changing the basic theory.

$\{e\}^A$	eth function computable with oracle A
$W_e^A = \text{dom}\{e\}^A$	eth r.e. set with oracle A

The constructions are verbatim the same. For example, a universal Turing machine turns into a universal Turing machine plus oracle.

The Use Principle

We continue to confuse a set $A \subseteq \mathbb{N}$ with its characteristic function. For any n write $A \upharpoonright n$ for the following finite approximation to the characteristic function:

$$(A \upharpoonright n)(z) \simeq \begin{cases} A(z) & \text{if } z < n, \\ \uparrow & \text{otherwise.} \end{cases}$$

For any such function α write $\alpha \sqsubset A$ if $\alpha = A \upharpoonright n$ for some n. Then

$$\{e\}^A(x) = y \iff \exists\, \alpha \sqsubset A \left(\{e\}^\alpha(x) = y\right)$$

with the understanding that the computation on the right never asks the oracle any questions outside of its domain.

Of course, a divergent computation may use the oracle infinitely often.

Definition

Let $A \subseteq \mathbb{N}$. The (Turing) jump of A is defined as

$$A' = K^A = \{\, e \mid \{e\}^A(e) \downarrow \}$$

So $\emptyset' = K^\emptyset = K$ is just the ordinary Halting set.

The nth jump $A^{(n)}$ is obtained by iterating the jump n times.

Lemma

A' is A-semidecidable but not A-decidable.

Proof.

The proof is verbatim the same as for the ordinary Halting problem.

□

So we get an infinite hierarchy of more and more complicated problems:

$$\emptyset,\ \emptyset',\ \emptyset'',\ \emptyset''',\ \ldots,\ \emptyset^{(n)},\ldots$$

In reality, though, nothing much beyond $\emptyset^{(4)}$ seems to play a role (except for problems that lie entirely outside of this hierarchy).

No End in Sight

- Even worse, we can collect all these sets into a single one, essentially by taking a disjoint union:

$$\emptyset^\omega = \{ \langle e, n \rangle \mid e \in \emptyset^{(n)} \}$$

- And nothing stops us from forming $(\emptyset^\omega)'$, $(\emptyset^\omega)''$ and so on. We can iterate the jump transfinitely often: ω^2 times, ω^ω times, $\varepsilon_0 = \omega^{\omega^{\omega^{\cdot^{\cdot^{\cdot}}}}}$ times.

- Headache, anyone?

Jump Properties

Let $A, B \subseteq \mathbb{N}$.

Theorem

- *A' is r.e. in A.*
- *A' is not Turing reducible to A.*
- *B is r.e. in A iff $B \leq_m A'$.*

Proof.

The first two parts are verbatim re-runs of the oracle-free argument.

Proof, Cont'd

For part (3) suppose B is r.e. in A. Hence there is a primitive recursive function f such that

$$\{f(x)\}^A(z) \simeq \begin{cases} 0 & \text{if } x \in B, \\ \uparrow & \text{otherwise.} \end{cases}$$

This function is A-computable since we can replace B on the right by W_e^A.

But then $x \in B \iff f(x) \in A'$, done.

Proof, Cont'd

Lastly suppose $B \leq_m A'$, say, $x \in B \iff f(x) \in A'$.

To enumerate B given A as oracle, proceed in stages

Stage s:

Compute $f(x)$ for all $x = 0, 1, \ldots, s-1$.

Enumerate A'_s (use oracle A for this).

If any of the $f(x)$ appear in A'_s, enumerate the corresponding x's into B.

Theorem

A is Turing equivalent to B iff A' is one-one-equivalent to B'.

Proof.

Assume $A \leq_T B$.

There is a primitive recursive function f such that

$$\{f(e)\}^B(z) \simeq \begin{cases} 0 & \text{if } \{e\}^A(z)\downarrow, \\ \uparrow & \text{otherwise.} \end{cases}$$

But then $e \in A' \iff f(e) \in B'$.

We can force f to be injective by doing something useless like counting to e first.

Proof, Cont'd

Now assume $A' \leq_1 B'$, say, $e \in A' \iff f(e) \in B'$.

There are a primitive recursive functions g_1, g_2 such that

$$\{g_1(e)\}^A(z) \simeq \begin{cases} 0 & \text{if } e \in A, \\ \uparrow & \text{otherwise.} \end{cases}$$

$$\{g_2(e)\}^A(z) \simeq \begin{cases} 0 & \text{if } e \notin A, \\ \uparrow & \text{otherwise.} \end{cases}$$

Now we combine f and the g_i to show how a TM with oracle B can determine membership in A.

Proof, Cont'd

$$
\begin{aligned}
e \in A &\iff g_1(e) \in A' \\
&\iff f(g_1(e)) \in B' \\
&\iff \{f(g_1(e))\}^B(f(g_1(e))) \downarrow \\
&\iff \exists\, \beta \sqsubset B\, \{f(g_1(e))\}^\beta(f(g_1(e))) \downarrow
\end{aligned}
$$

Likewise

$$
e \notin A \iff \exists\, \beta \sqsubset B\, \{f(g_2(e))\}^\beta(f(g_2(e))) \downarrow
$$

Since one of the two computations must converge we can decide which with oracle B.

The Jump Hierarchy

One writes $\emptyset^{(n)}$ for the Turing degree of of $\emptyset^{(n)}$. Then

- $\emptyset$ is the degree of all decidable sets.
- $\emptyset'$ is the degree of K.
- $\emptyset''$ is the degree of FIN and TOT.
- $\emptyset'''$ is the degree of REC.

A Horror

Nothing stops us from constructing

$$\emptyset^\omega = \{ \langle n, x \rangle \mid x \in \emptyset^{(n)} \}$$

This is essentially the ω-iterate of the jump on $\emptyset$. Note that

$$\emptyset <_T \emptyset' <_T \emptyset'' <_T \ldots <_T \emptyset^{(n)} <_T \ldots <_T \emptyset^\omega$$

And, of course, we could apply the jump to $\emptyset^\omega$.

We won't go there.

Jump Theorem

One might wonder which degrees are of the form $\boldsymbol{a}'$.

Theorem (Friedberg)

Let $\boldsymbol{a} \geq_T \emptyset'$. Then $\boldsymbol{a} = \boldsymbol{b}'$ for some $\boldsymbol{b}$.

In other words, the range of the jump operator is the cone $\boldsymbol{x} \geq_T \emptyset'$.

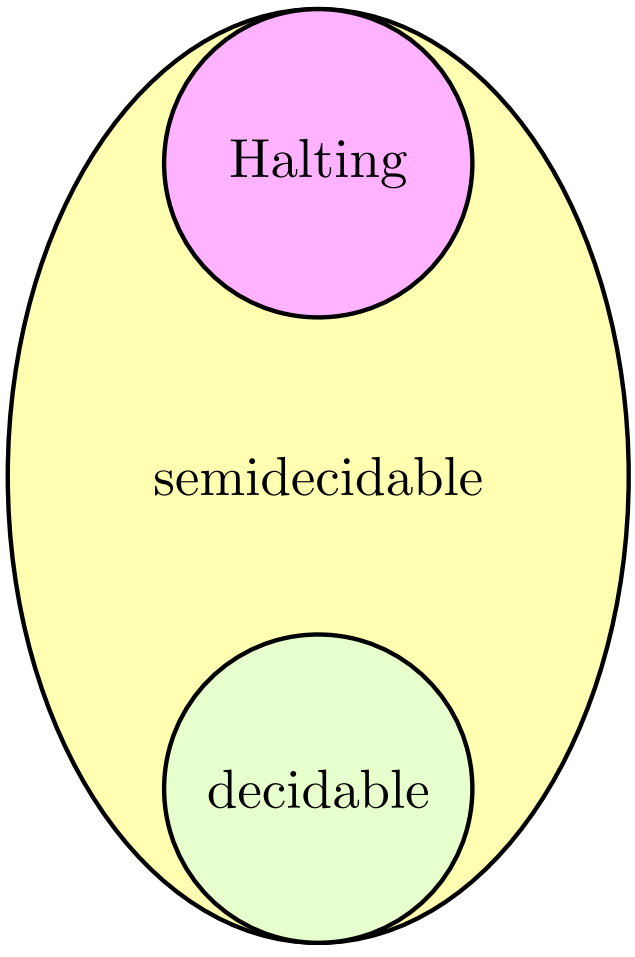
1 The Turing Jump

2 Arithmetical Hierarchy

3 Definability

4 Formal Systems

Recall: The World



co-semidecidable

A Geometric Approach

Logically, the step from decidable (or even primitive recursive) to semidecidable corresponds to unbounded search, or existential quantification.

The step from semidecidable to co-semidecidable is negation.

Here is an idea: these logical operations have geometric counterparts: negation corresponds to complements, and existential quantification corresponds to projections.

What happens if we apply projections and complements systematically to decidable sets?

Projections and Complements

Let's write down careful definitions for these purely set-theoretic operations (which might appear to have nothing to do with computability).

Definition

Let $A \subseteq \mathbb{N} \times \mathbb{N}^n$ where $n \geq 1$. The projection of A is the set

$$\text{proj}(A) = \{\, \boldsymbol{x} \in \mathbb{N}^n \mid \exists\, z\, (z, \boldsymbol{x}) \in A \,\} \subseteq \mathbb{N}^n.$$

The complement of A is understood to be $\mathbb{N}^n - A$.

For any collection $\mathcal{C} \subseteq \mathfrak{P}(\mathbb{N}^n)$ of subsets of $\mathbb{N}^n$, $n \geq 1$, define $\text{proj}(\mathcal{C})$ to be the collection of all projections of sets in $\mathcal{C}$. Likewise, define $\text{compl}(\mathcal{C})$ to be the collection of all complements of sets in $\mathcal{C}$.

Arithmetical Hierarchy

Definition

Define classes of subsets of $\mathbb{N}^n$:

$$\Sigma_0 = \Pi_0 = \text{ all decidable sets}$$

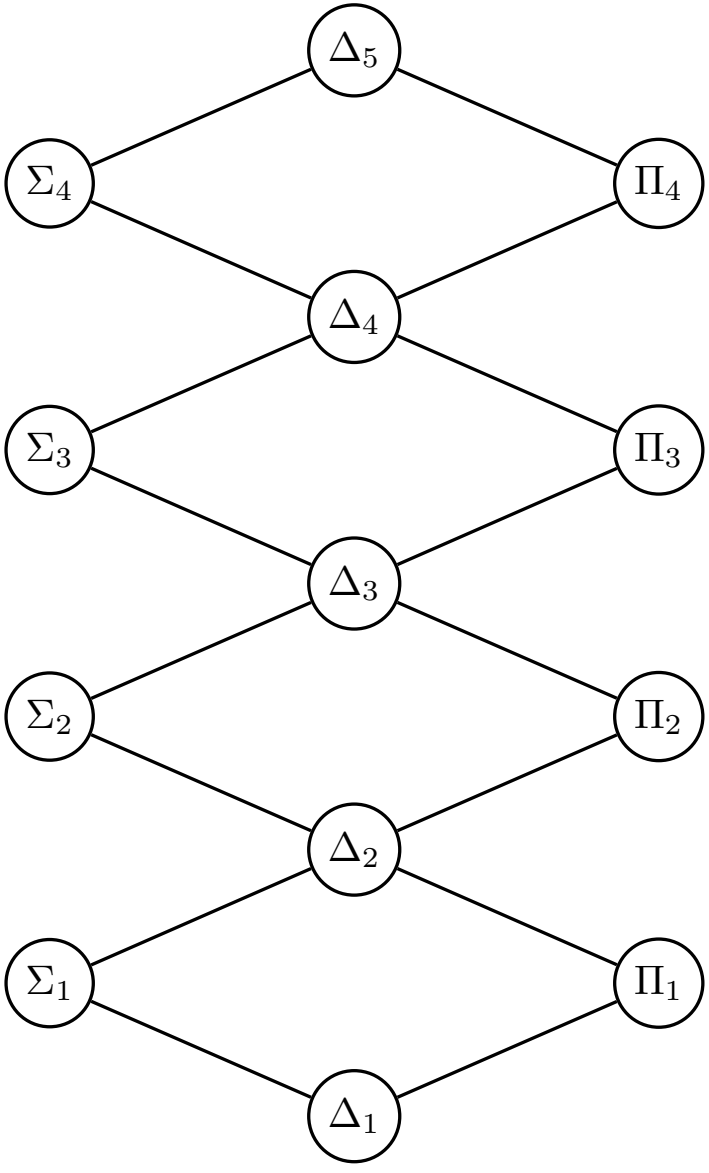
$$\Sigma_{k+1} = \text{proj}(\Pi_k)$$

$$\Pi_k = \text{compl}(\Sigma_k)$$

$$\Delta_k = \Sigma_k \cap \Pi_k$$

Thus, Δ_1 is the class of all decidable sets, Σ_1 is the class of all semidecidable sets, and Π_1 is the class of all co-semidecidable sets.

A Diagram


Δ_5
Σ_4
Π_4
Δ_4
Σ_3
Π_3
Δ_3
Σ_2
Π_2
Δ_2
Σ_1
Π_1
Δ_1

But Does Anyone Really Live There?

Consider the primitive recursive set $R \subseteq \mathbb{N}^3$: $R(s, x, e)$ if $\{e\}$ on x converges in at most s steps.

project

$$(x, e) \in \mathsf{proj}(R) \iff \{e\} \text{ on } x \text{ converges}$$

complement

$$(x, e) \in \overline{\mathsf{proj}(R)} \iff \{e\} \text{ on } x \text{ diverges}$$

project

$$e \in \mathsf{proj}(\overline{\mathsf{proj}(R)}) \iff \{e\} \text{ diverges on some } x$$

complement

$$e \in \overline{\mathsf{proj}(\overline{\mathsf{proj}(R)})} \iff \{e\} \text{ is total}$$

Total Functions

This shows that our set

$$\mathsf{TOT} = \{\, e \in \mathbb{N} \mid \{e\} \text{ total} \,\}$$

of total computable function is Π_2.

Instead of performing projections and complementations, this is easier to see by counting quantifiers:

$$e \in \mathsf{TOT} \iff \forall\, x\, \exists\, s\, R(s, x, e).$$

Make sure you understand how this logical characterization corresponds to the previous slide.

Arithmetical Sets

Definition

A set $A \subseteq \mathbb{N}^n$ is arithmetical if it belongs to some class Σ_k.

We note in passing that the class of arithmetical sets is countable, so it follows by cardinal arithmetic that we are missing most subsets of $\mathbb{N}$.

One interesting example of a set we are missing is arithmetical truth: the set of all (code numbers of) formulas of arithmetic that are true is not an element in this hierarchy (arithmetic truth a so-called Δ_1^1 set, just outside of our arithmetical sets).

The Bottom of AH

Of practical relevance are mostly the first few levels of the arithmetical hierarchy. We have already seen examples of decidable, semidecidable sets and co-semidecidable sets.

Here are some other examples.

- TOT is Π_2.
- The indices of all finite r.e. sets form a Σ_2 set:

$$\mathsf{FIN} = \{\, e \in \mathbb{N} \mid W_e \text{ finite} \,\}$$

- The indices of all cofinite r.e. sets form a Σ_3 set:

$$\mathsf{Cof} = \{\, e \in \mathbb{N} \mid W_e \text{ cofinite} \,\}$$

None of these sets belong to the next lower Δ_k level of the hierarchy.

Exercises

Exercise

Show that FIN *is* Σ_2 *by constructing this set using projections and complementation.*

Exercise

Show that Cof *is* Σ_3 *by constructing this set using projections and complementation.*

Exercise

Find the position in the hierarchy of

$$\mathsf{INF} = \{\, e \in \mathbb{N} \mid W_e \text{ infinite} \,\}.$$

A Σ_3 Property

Recall that $W_e = \mathrm{dom}\{e\}$ is the eth c.e. set.

How hard is it to check, given the index e, whether W_e is decidable?

As we have seen,

$$W_e \text{ decidable} \iff \exists\, e' \,(W_e \cap W_{e'} = \emptyset \wedge W_e \cup W_{e'} = \mathbb{N})$$

The second condition in the formula is Π_2 and the first is Π_1.

Hence the following set is Σ_3:

$$\mathrm{REC} = \{\, e \in \mathbb{N} \mid W_e \text{ decidable} \,\}$$

Exercise

Carefully check that $\mathcal{R}$ is indeed Σ_3.

A Σ_4 Property

It is difficult to find natural examples higher up in the hierarchy.

Here a last one: the indices of all complete c.e. sets form a Σ_4 set:

$$\text{Comp} = \{\, e \in \mathbb{N} \mid \text{domain of } \{e\} \text{ is complete} \,\}$$

For $e \in \text{Comp}$ we need: there is a reduction $\{\widehat{f}\}$ from K to W_e.

Exercise

Carefully check that Comp *is indeed* Σ_4*.*

But Is It Real?

So there are several problems that appear at the bottom levels of the AH. We also know that $\Delta_1 \subsetneq \Sigma_1, \Pi_1$.

But how about higher up? For example, we don't yet know that TOT $\notin \Delta_2$.

Maybe $\Delta_2 = \Sigma_k = \Pi_k$ for all $k \geq 2$ (a so-called collapsing hierarchy)?

Completeness

Let's call a set Σ_k-complete if it is one-one complete for the class of Σ_k sets, and likewise for Π_k.

So K is Σ_1-complete and $\mathbb{N} - K$ is Π_1-complete.

In a moment we will use Σ_k-complete sets to show that the levels of the hierarchy are indeed distinct.

First a bit more background about these classes.

Characterizing Σ_k

Lemma

$A \in \Sigma_{k+1}$ iff A is r.e. in B for some $B \in \Pi_k \cup \Sigma_k$.

Proof.

By definition $A \in \Sigma_{k+1}$ if there is some $B \in \Pi_k$ such that

$$x \in A \iff \exists\, z\, B(z, x)$$

so that A is r.e. in B.

If A is r.e. in B we have $A = \mathsf{dom}\{e\}^B$. Let

$$B_0 = \{\, (s, x) \mid \{e\}^B_s(x) \simeq 0 \,\}.$$

The B_0 is Π_k and $A = \mathsf{proj} B_0$.

Oracles are invariant under complementation, so the result also follows for $B \in \Sigma$.

□

Theorem

$\emptyset^{(k)}$ is Σ_k-complete for $k > 0$.

Proof. By induction on k, $k = 1$ is clear.

So assume $\emptyset^{(k)}$ is Σ_k-complete and $A \in \Sigma_{k+1}$.

Then A is B-r.e. for some B in Σ_k by the lemma.

By IH A is also $\emptyset^{(k)}$-r.e.

This implies $A \leq_1 \emptyset^{(k+1)}$ by the jump theorem.

□

Another Characterization for Σ_k

Lemma

$A \in \Sigma_{k+1}$ iff A is r.e. in $\emptyset^{(k)}$.
Likewise, $A \in \Delta_{k+1}$ iff $A \leq_T \emptyset^{(k)}$.

Proof.

Part (1) follows from the last theorem.

For part (2) note that $A \in \Delta_{k+1}$ iff both A and $\mathbb{N} - A$ are r.e. in $\emptyset^{(k)}$ by part (1).

The latter condition means both $A \leq_T \emptyset^{(k)}$.

□

Corollary

All the inclusions $\Delta_k \subsetneq \Sigma_k, \Pi_k \subsetneq \Delta_{k+1}$ are proper, $k \geq 1$.

Proof.

We know that $\emptyset^{(k)}$ is Σ_k-complete.

Assume for a contradiction that $\emptyset^{(k)} \in \Pi_k$ so that $\emptyset^{(k)} \in \Delta_k$.

By the last lemma $\emptyset^{(k)} \leq_T \emptyset^{(k-1)}$, clearly a contradiction.

□

And Examples?

The hierarchy is proper, and we have seen examples of natural sets that appear the first few levels. Alas, thing get murky fairly quickly a few levels higher up.

For example, to produce an example for a Π_5 set we could ask whether a computable function f enumerates only indices of universal machines:

$$\forall\, x \left(f(x) \text{ is a UTM} \right)$$

This is arguably less compelling than, say, TOT.

1 The Turing Jump

2 Arithmetical Hierarchy

3 **Definability**

4 Formal Systems

The Natural Numbers

Arithmetic takes place in the structure of natural numbers:

$$\mathcal{N} = \langle \mathbb{N}, +, *, <, 0, 1 \rangle$$

Since computability is classically explained in terms of arithmetical functions (functions on the natural numbers, as opposed to functions on words which are more natural in complexity theory) one might wonder what the connection between $\mathcal{N}$ is.

The built-in functions addition and multiplication as well as the natural order relation are all primitive recursive. Where are the computable functions?

A Language

Let us fix a small language $\mathcal{L}$ for $\mathcal{N}$: first-order logic language with function/relation symbols for all the built-ins.

A bounded quantifier is a quantifier of the form

$$\exists\, x < z\, \varphi(x, z)$$
$$\forall\, x < z\, \varphi(x, z)$$

z is a free variable here, x is bound.

Definition

A formula of $\mathcal{L}$ is Δ_0 if it does not contain any unbounded quantifiers.

Example: Primality

$$x \operatorname{div} y = \exists\, z \leq y\, x * z = y$$
$$P(x) = \forall\, z \leq x\, (z \operatorname{div} x \to z = 1 \vee z = x) \wedge x > 1$$

Then

$$\{\, n \in \mathbb{N} \mid \mathcal{N} \models P(n) \,\}$$

is the set of all prime numbers.

Hence the primes can be defined by a Δ_0 formula.

Definability

In general, given a formula $\varphi(x_1, \ldots, x_n)$ with n free variables as indicated defines a set

$$\varphi^{\mathcal{N}} = \{\, (a_i, \ldots, a_n) \in \mathbb{N}^n \mid \mathcal{N} \models \varphi(a_1, \ldots, a_n) \,\}$$

Δ_0 and Primitive Recursive

Lemma

For any Δ_0 formula φ, the set $\varphi^{\mathcal{N}}$ is primitive recursive.

Proof.

Induction on the buildup of φ.

This is clear for atomic formulae such as $x < y$, $x * y = z$ and so forth.

Logical connectives $\wedge$, $\vee$ and $\neg$ are OK since primitive recursive relations are closed under union, intersection and complement.

Bounded quantifiers can be handled by bounded search, see the notes on primitive recursive functions.

□

Primitive Recursive and Δ_0

Lemma

For any primitive recursive set $A \subseteq \mathbb{N}$ there is a Δ_0 formula φ such that $\varphi^{\mathcal{N}} = R$.

Proof.

Induction on the definition of A (or rather, the primitive recursive function that defines A).

Obviously the basic primitive recursive functions are Δ_0-definable.

Δ_0-definable functions are closed under composition:

$$\exists\, y\, (\varphi(x, y) \wedge \psi(y, z))$$

□

Σ_k and Π_k Formulae

A formula is Σ_0 or Π_0 if it is Δ_0.

A formula is Σ_{k+1} if it is obtained from a Π_k formula by prefixing it with a block of existential quantifiers.

A formula is Π_{k+1} if it is obtained from a Σ_k formula by prefixing it with a block of universal quantifiers.

For example,

$$\forall\, x\, \exists\, y_1, y_2\, \varphi(x, y_1, y_2, z)$$

is Π_2 assuming φ is Δ_0.

A set $A \subseteq \mathbb{N}$ is Σ_1-definable if for some Σ_1 formula φ we have

$$A = \varphi^{\mathcal{N}}$$

So this means

$$a \in A \iff \exists\, x_1, \ldots, x_k\, \varphi(x_1, \ldots, x_k, a)$$

where φ is Δ_0.

Σ_1 Is Semidecidable

Theorem

A set $A \subseteq \mathbb{N}$ is Σ_1-definable iff it is semidecidable.

Sketch of proof. Given a Σ_1 formula, we can perform an unbounded search for a witness.

Given a semidecidable set, we can express it's definition in terms of, say, Gödel-Herbrand equations as an arithmetic formula, using the standard coding machinery. The existence of a derivation can be handled by a single existential quantifier.

□

Σ_k and Π_k Sets

A set $A \subseteq \mathbb{N}$ is Σ_n-definable if for some Σ_n formula φ we have $A = \varphi^{\mathcal{N}}$.

Likewise for Π_n-definable.

A set $A \subseteq \mathbb{N}$ is Δ_n-definable if is Σ_n-definable and Π_n-definable.

Proposition

A set $A \subseteq \mathbb{N}$ is Σ_n-definable iff its complement is Π_n-definable.

Corollary

The decidable sets are exactly the Δ_1-sets.

Quantifiers

Since projection corresponds to existential quantification, and complementation corresponds to negation, we can characterize the arithmetical hierarchy as follows.

Lemma

A set $A \subseteq \mathbb{N}$ is Σ_k where $k \geq 1$ if, and only if, there is a decidable relation $R \subseteq \mathbb{N} \times \mathbb{N}^k$ such that

$$a \in A \iff \exists\, x_1 \,\forall\, x_2 \,\ldots Q\, x_k \, R(a, x_1, \ldots, x_k).$$

An analogous result holds for Π_k and for $A \subseteq \mathbb{N}^n$.

This is the logical, definability-theoretic version of the arithmetical hierarchy as opposed to the set-operation definition from above. One nice feature of this version is that it generalizes easily to other contexts (complexity theory, generalized computability theory).

Arithmetization

The intended meaning of

$$\exists\, x_1 \,\forall\, x_2 \;\ldots Q\, x_k \, R(a, x_1, \ldots, x_k).$$

is of course that the variables x_i range over $\mathbb{N}$. Thus we have a definition of A over the structure

$$\langle \mathbb{N}, R \rangle$$

where $R \subseteq \mathbb{N}^{k+1}$ is decidable.

Since R changes with each arithmetical set A this is not too useful a description.

More interesting would be a definition of the standard structure of arithmetical, the natural numbers with the usual operations:

$$\mathfrak{N} = \langle \mathbb{N}, +, \cdot, S, 0, 1, < \rangle.$$

Gödel's Definition

In fact, in 1931 Gödel defined a set $A \subseteq \mathbb{N}^n$ to be arithmetical if it could be defined using addition and multiplication in first order logic with equality. Thus, arithmetical in the sense of Gödel means definable over

$$\mathfrak{N}_{+,\cdot} = \langle \mathbb{N}, +, \cdot \rangle$$

Note that all the missing operations, constants and relations are easily definable in terms of addition and multiplication, so there is no loss in moving to the smaller structure – except that the definitions may become slightly more complicated.

Exercise

Show how to define constants 0 and 1, the successor function and the order relation in $\mathfrak{N}_{+,\cdot}$.

Defining Functions

To see the equivalence of Gödel's approach, let us consider definitions of number-theoretic functions.

Definition

A relation $A \subseteq \mathbb{N}^n$ is definable in elementary arithmetical if there is a formula φ in the language of arithmetical such that

$$\boldsymbol{a} \in A \iff \mathfrak{N} \models \varphi(\underline{\boldsymbol{a}})$$

A function is so definable if its graph is:

$$f(\boldsymbol{a}) = b \iff \mathfrak{N} \models \varphi(\underline{\boldsymbol{a}}, \underline{b})$$

The appearance of numerals $\underline{a}$ to represent the natural number a in the defining formula is a nuisance but one should distinguish between syntax and semantics at this point: 3 is a natural number, but it is represented by the formal term $S(S(S(0)))$.

Δ_0 Definitions

Arbitrary formulae in arithmetical are much too complicated for computational purposes; we cannot hope to cope with long blocks of alternating quantifiers. Here is a more modest class of formulae.

Definition

A formula of arithmetical is Δ_0 if it is formed from atomic formulae using only logical connectives and bounded quantifiers.
A relation/function is Δ_0-definable if the formula φ above can be chosen to be Δ_0.

In other words, unbounded quantifiers are not allowed. For example,

$$R(a) = a > 1 \wedge \forall\, x < a\, (\neg \exists\, y < a\, (x \cdot y \approx a))$$

provides a Δ_0 definition of the primes.

Note that any Δ_0-definable relation is automatically decidable.

Defining Computable Functions

For the opposite direction we need more than Δ_0.

Clearly, the basic functions constants, projections, addition and multiplication are all Δ_0-definable.

However, if we want to compose computable functions we need an existential quantifier: the intermediate value can only be found by unbounded search.

$$(f \circ g)(x) = y \iff \exists\, z\, (g(x) = z \wedge f(z) = y).$$

Unbounded search in the form of a min operation requires a bounded universal quantifier. Let $f(x) = \min\big(\, z \mid g(z, x) = 0 \,\big)$. Then

$$f(x) = y \iff g(y, x) = 0 \wedge \forall\, z < y\, g(z, x) \neq 0.$$

Existential Formulae

It follows that for any computable function f we have

$$f(x) = y \iff \mathfrak{N} \models \exists\, z_1, \ldots, z_n\, \varphi(\boldsymbol{z}, x, y)$$

where φ is Δ_0.

With more effort one can use the sequence number machinery to collapse all the existential quantifiers into a single one:

$$f(x) = y \iff \mathfrak{N} \models \exists\, z\, \varphi(z, x, y).$$

Similarly, every semidecidable relation $A \subseteq \mathbb{N}$ has a definition

$$x \in A \iff \mathfrak{N} \models \exists\, z\, \varphi(z, x).$$

Arithmetical is Definable

If we consider definitions with arbitrarily many quantifiers we obtain all sets in the arithmetical hierarchy.

Theorem

A relation is arithmetical if, and only if, it is definable in elementary arithmetical.

It should be noted that arithmetical truth itself is not arithmetical: the assertion "φ is a valid sentence of arithmetical" cannot be described by a formula of arithmetical.

Here we assume some standard Gödel style coding of formulae. Also note that validity is used in the old-fashioned way: true over the natural numbers (not: over all models).

1 The Turing Jump

2 Arithmetical Hierarchy

3 Definability

4 **Formal Systems**

So computability translates into easily definable over $\mathfrak{N}$: to determine the value of a computable function we have to check the truth of a Σ_1 formula.

Likewise, to check membership in a semidecidable set we have to check the truth of a Σ_1 formula.

Again, the difference between semidecidable and decidable all boils down to a single unbounded search: we cannot bound the witness z in $\exists\, z\, \varphi(z, x)$.

How about provability (as opposed to truth)?

How hard is to prove that $f(a) = b$ for a computable function?

Obviously, the computation of the output constitutes some kind of proof (provided the calculation terminates): we can easily check that no errors were made in the computation. But we want a classical proof in some formal system of arithmetical.

As usual, we assume a suitable language of arithmetical, some sublanguage of $\mathcal{L}(+, \cdot, S, 0, 1, <)$.

A standard choice is Peano Arithmetical: axioms for the successor function, primitive recursive definitions of addition and multiplication plus and induction axiom (actually, a schema).

The challenge is to determine whether

$$f(a) = b \iff (\mathrm{PA}) \vdash \varphi(\underline{a}, \underline{b}).$$

when f is computable.

As we will see, one can actually get away with much less; there is a surprisingly tiny system of arithmetical comprised of just 7 axioms (no schemata at all) dealing with successor, addition and multiplication.

Numerals

Sine we are dealing with arbitrary arithmetical theories we have to be a bit careful about representations of natural numbers. Our theories have a constant 0 (and we may safely assume that there is a constant 1).

But, 5 won't be constant in the language. However, we can easily represent 5:

$$\underline{5} = S(S(S(S(S(0))))).$$

This notation is too cumbersome to use in the real world, but it is good enough for our purposes. More elegant solutions would make it necessary to increase the complexity of the language quite a bit.

Representability

Suppose T is some theory of arithmetical. We need to explain precisely what it means for T to express facts about some function f.

Definition

A function $f : \mathbb{N}^n \to \mathbb{N}$ is representable in T if there is a formula φ such that

$$f(\boldsymbol{a}) = b \iff T \vdash \forall z\,(\varphi(\underline{\boldsymbol{a}}, z) \leftrightarrow z \approx \underline{b}).$$

In this case $\varphi(\boldsymbol{x}, z)$ represents f.

Hence, for f to be representable we need in particular

$$T \vdash \varphi(\underline{\boldsymbol{a}}, \underline{b}).$$

But, this is not enough: we also need uniqueness:

$$T \vdash \varphi(\underline{\boldsymbol{a}}, a) \wedge \varphi(\underline{\boldsymbol{a}}, v) \to u \approx v.$$

Simple Examples

For example, addition and pairing are represented in any sane theory of arithmetical by the formulae

$$\varphi(x_1, x_2, z) = x_1 + x_2 \approx z$$
$$\varphi(x_1, x_2, z) = (x_1 + x_2) \cdot (x_1 + x_2 + \underline{1}) \approx \underline{2} \cdot z.$$

Of course, these are cheap shots: we can explicitly write down a term in the language that represents the function, so

$$\varphi(\boldsymbol{x}, z) = t(\boldsymbol{x}) \approx z$$

or a slightly more complicated right hand side. Note that all polynomials can be got this way.

But how about functions not expressed by terms?

Less Simple Examples

Predecessor, exponentiation, GCD and so forth come to mind.

The predecessor function is not too bad:

$$\varphi(x, z) = (x \approx 0 \wedge z \approx 0) \vee \exists\, y\, (x \approx S(y) \wedge z \approx y).$$

Note that we can bound the quantifier, so there is a Δ_0 representation for predecessor.

But exponentiation is a real problem: the standard p.r. definition cannot be directly translated into a formula of arithmetical. We would need to mention φ on the right hand side, which is of course strictly verboten.

Exercise

Show that the GCD function is representable (say, in Peano Arithmetical).

Totality

It is important to distinguish between representability of a function f in some theory and totality: we are considering number-theoretic functions $f : \mathbb{N}^n \to \mathbb{N}$ which are automatically total.

But there is no requirement for T to prove anything about totality; we do not insist that

$$T \vdash \forall \boldsymbol{x} \exists y \, \varphi(\boldsymbol{x}, y).$$

This may seem a bit unnatural, but as it turns out proofs of totality are quite hard.

There are many computable functions whose totality cannot be proven in a fairly powerful system such as Peano Arithmetical. Of course, any real-world computable function can be proven total in (PA).

Closure Properties

We need some closure properties for representable functions.

Composition is easy. Here is the case of composition of two unary functions, represented by φ_1 and φ_2, respectively.

$$\varphi(x, z) = \exists\, y\, (\varphi(x, y) \wedge \varphi(y, z))$$

Note that the quantifier here is unbounded: we cannot predict how far we have to search to find the value for the first function in the composition.

Exercise

Produce a representation for arbitrary composition.

Exercise

Verify that the definition really works in (PA).

More Closure

It is tempting to try to establish closure under primitive recursion, but that's rather tedious. A better way is to show closure under regular search and then show in general that regular search is enough to get primitive recursion (see the notes on Primitive Recursive Functions).

So suppose $g : \mathbb{N}^{k+1} \to \mathbb{N}$ is a function represented by ψ such that

$$\forall\, \boldsymbol{a}\, \exists\, b\, g(b, \boldsymbol{a}) = 0.$$

Then $f = \min g$ is represented by

$$\varphi(\boldsymbol{x}, z) = \psi(z, \boldsymbol{x}, 0) \wedge \forall\, y < z\, \neg\psi(y, \boldsymbol{x}, 0).$$

The right hand side forces z to be the least argument for which g returns 0.

Computable is Representable

Theorem

A number-theoretic function is computable if, and only if, it is representable in Peano Arithmetical.

So Peano's axioms are powerful enough to express all possible computations. This may not be terribly surprising, but it turns out that nowhere near the full power of (PA) is needed in order to get representations of all computable functions.

Here is an amazingly tiny system due to Robinson.

Robinson's System Q

successor

$$S(x) \not\approx 0 \qquad S(x) \approx S(y) \to x \approx y$$

addition

$$x + 0 \approx x \qquad x + S(y) \approx S(x + y)$$

multiplication

$$x \cdot 0 \approx 0 \qquad x \cdot S(y) \approx (x \cdot y) + x$$

weak induction

$$x \not\approx 0 \to \exists\, y\, S(y) \approx x$$

What is called "weak induction" here is nothing but the assertion that the range of the successor function is everything except for 0.

Theorem

System Q represents all computable functions.

This is truly amazing and somewhat tedious to prove, as one might imagine. In many regards, Q is really much to small to axiomatize arithmetical.

Exercise

Show that none of the following theorems of arithmetical are provable in Q: $x \neq S(x)$, $x + (y + z) \approx (x + y) + z$, $x + y \approx y + x$.

Summary

- Church's Thesis states that Turing computability precisely captures the intuitive notion of computability.
- For practical algorithms, one needs a much more fine grained analysis based on strict resource bounds.
- In the absence of hard lower bounds, hardness and completeness are helpful to compare the difficulty of problems.
- Alas, life becomes much harder in low complexity classes such as $\mathbb{P}$ and $\mathbb{NP}$.