

# CDM

## Arithmetical Hierarchy

Klaus Sutner  
Carnegie Mellon University  
www.cs.cmu.edu/~sutner

### Battleplan

- Beyond Undecidability
- Arithmetical Hierarchy
- Definability
- Representability

### The World of Complexity

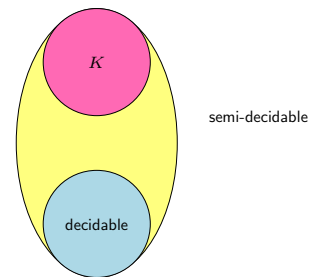
So far we have the following rough classification of decision problems in terms of abstract computability:

- Easily solvable: primitive recursive
- Solvable in principle: decidable
- Unsolvable in principle: undecidable

Take the “easily” with a grain of salt, we are not concerned with efficient computation yet. We will have to undertake a much more careful analysis in a while, see the notes on Complexity Theory.

But before we do that, just a few more comments on undecidable problems. As it turns out, the arguments are easier to understand in the world of unconstrained computability theory than in complexity theory. The additional resource bounds make like much more difficult, and unsolved problems abound in complexity theory.

### The Classical World of Complexity



Here  $K$  is the Halting Problem, the standard example of a complete semi-decidable problem.

### Beyond Semi-Decidable

We have seen a few decidable and semi-decidable (but undecidable) decision problems.

The question arises: Are there any other types of problems?

The answer is already implicit in a previous lemma that links decidability and semi-decidability.

**Lemma 1.** *The complement of any semi-decidable but not decidable set is not semi-decidable.*

*Proof.* Suppose  $A$  is semi-decidable. If  $\mathbb{N} - A$  were also semi-decidable, then, by a previous lemma,  $A$  would be decidable.  $\square$

The complements of semi-decidable sets are sometimes called *co-semi-decidable*.

### Example: Halting

A typical example is the Halting Set  $K$ .

There is a striking difference between  $K$  and its complement:

$$e \in K \iff \exists t \text{ trace}(e, e, t)$$
$$e \notin K \iff \forall t \neg \text{trace}(e, e, t)$$

where trace is decidable, even primitive recursive.

For  $K$  we can semi-decide membership, but for  $\mathbb{N} - K$  we can only semi-decide non-membership.

So there is a strange asymmetry between a problem and its negation that arises when we consider semi-decidable problems.

By contrast, for decidable problems, negation is irrelevant.

This asymmetry is a standard stumbling block in the understanding of complexity classes such as NP.

## Projections and Complements

We can construct difficult sets by using purely set-theoretic operations (which appear to have nothing much to do with computability).

**Definition 1.** Let  $A \subseteq \mathbb{N}^n$  where  $n \geq 2$ . The **projection** of  $A$  is the set

$$\text{proj}(A) = \{ \vec{x} \in \mathbb{N}^{n-1} \mid \exists z (z, \vec{x}) \in A \} \subseteq \mathbb{N}^{n-1}.$$

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

By the Kleene normal form theorem, every semi-decidable set is a projection of a decidable set.

Complementation of a semi-decidable set produces a co-semi-decidable set, which, as we have seen, may well fail to be semi-decidable.

What happens if we keep projecting and taking complements?

## Total Recursive Functions

Here is a typical example. Define the set  $R \subseteq \mathbb{N}$

$$R(s, t, e) \iff \{e\}(t) \text{ halts after at most } s \text{ steps.}$$

$R$  is primitive recursive and thus in particular decidable.

If we project, complement, project and complement again we get

$$\text{TOT} = \{ e \in \mathbb{N} \mid \{e\} \text{ is a total function,} \}$$

the collection of all (indices of) total recursive functions.

Rewritten with quantifiers we have

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

Thus, to decide membership in TOT we would have to conduct an unbounded search for each value of  $t$ , a kind of nested infinite computation.

## Even Magic Fails

One can show that TOT is neither semi-decidable nor co-semi-decidable.

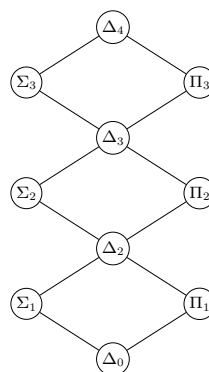
In fact, suppose the famous magician Chidur offers you a magic KBox<sup>TM</sup> that can decide membership in  $K$ .

The way it works is secret and unknowable, but it is guaranteed to always give the right answer, instantaneously. Supposedly there is a black hole inside, but no one really knows.

Then even with the help of this magic box it would still be impossible to decide membership in TOT: we cannot build a decision algorithm for TOT, even when this algorithm is allowed to call the box as a subroutine.

This kind of argument is very useful to compare the relative difficulty of problems, even when they are undecidable or have no (known) efficient algorithms. It is also used a lot in complexity theory.

## Arithmetical Hierarchy



The first few levels of the hierarchy. All inclusions are proper.

## Arithmetical Hierarchy

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

**Definition 2.** Define classes of subsets of  $\mathbb{N}^n$ :

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

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

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

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

where  $k \geq 1$ .

Thus,  $\Delta_1$  is the class of all decidable sets,  $\Sigma_1$  is the class of all semi-decidable sets, and  $\Pi_1$  is the class of all co-semi-decidable sets.

## A Proper Hierarchy

One can show that this hierarchy is proper:

**Theorem 1.**

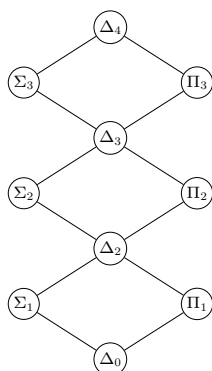
$$\Sigma_k, \Pi_k \subsetneq \Delta_{k+1} \subsetneq \Sigma_{k+1}, \Pi_{k+1}.$$

The argument is quite similar to the one used to show that the Halting Problem is semi-decidable ( $\Sigma_1$  in the hierarchy) but not decidable ( $\Delta_0$  in the hierarchy).

**Definition 3.** A set  $A \subseteq \mathbb{N}^n$  is **arithmetical** if it belongs to some class  $\Sigma_k$ .

We note in passing that the class of arithmetical sets is countable, so it follows by Cantor-style brute force that we are missing most subsets of  $\mathbb{N}$ . For example, arithmetical truth is not an element in this hierarchy. But many, many real-world sets are.

### A Proper Hierarchy



The first few levels of the hierarchy. All inclusions are proper.

### The Bottom of AH

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

Here are some other examples.

- TOT is  $\Pi_2$ .
- The indices of all finite c.e. sets form a  $\Sigma_2$  set:

$$\text{FIN} = \{ e \in \mathbb{N} \mid \text{domain of } \{e\} \text{ is finite} \}$$

- The indices of all cofinite c.e. sets form a  $\Sigma_3$  set:

$$\text{Cof} = \{ e \in \mathbb{N} \mid \text{domain of } \{e\} \text{ is cofinite} \}$$

- The indices of all complete c.e. sets form a  $\Sigma_4$  set:

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

None of these sets belong to the next lower  $\Delta_k$  level of the hierarchy.

### Exercises

**Exercise 1.** Show that TOT is  $\Pi_2$  by constructing this set using projections and complementation.

**Exercise 2.** Show that Cof is  $\Sigma_3$  by constructing this set using projections and complementation.

**Exercise 3.** Find the position in the hierarchy of

$$\text{INF} = \{ e \in \mathbb{N} \mid \text{domain of } \{e\} \text{ is infinite} \}.$$

### A $\Sigma_3$ Property

Recall that  $W_e = \text{dom } \{e\}$  is the  $e$ th 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  $\Pi_2$  and the first is  $\Pi_1$ .

Hence the set

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

is  $\Sigma_3$ .

**Exercise 4.** Carefully check that REC is indeed  $\Sigma_3$ .

### Quantifiers

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

**Lemma 2.** A set  $A \subseteq \mathbb{N}$  is  $\Sigma_k$  where  $k \geq 1$  if, and only if, there is a decidable relation  $R \subseteq \mathbb{N} \times \mathbb{N}^k$  such that

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

An analogous result holds for  $\Pi_k$  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

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

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 arithmetic, 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 5.** 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 4.** A relation  $A \subseteq \mathbb{N}^n$  is **definable in elementary arithmetic** if there is a formula  $\varphi$  in the language of arithmetic such that

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

A function is so definable if its graph is:

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

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  $S(S(S(0)))$ .

## $\Delta_0$ Definitions

Arbitrary formulae in arithmetic 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 5.** A formula of arithmetic is  $\Delta_0$  if it is formed from atomic formulae using only logical connectives and bounded quantifiers.

A relation/function is  $\Delta_0$ -definable if the formula  $\varphi$  above can be chosen to be  $\Delta_0$ .

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  $\Delta_0$  definition of the primes.

Note that any  $\Delta_0$ -definable relation is automatically decidable.

## Defining Computable Functions

For the opposite direction we need more than  $\Delta_0$ .

Clearly, the basic functions constants, projections, addition and multiplication are all  $\Delta_0$ -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  $f(x) = \min(z \mid g(z, x) = 0)$ . 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

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

where  $\varphi$  is  $\Delta_0$ .

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 semi-decidable relation  $A \subseteq \mathbb{N}$  has a definition

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

## Arithmetic is Definable

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

**Theorem 2.** A relation is arithmetical if, and only if, it is definable in elementary arithmetic.

It should be noted that arithmetic truth itself is not arithmetical: the assertion “ $\varphi$  is a valid sentence of arithmetic” cannot be described by a formula of arithmetic.

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).

## Computability, Truth and Provability

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  $\Sigma_1$  formula.

Likewise, to check membership in a semi-decidable set we have to check the truth of a  $\Sigma_1$  formula.

Again, the difference between semi-decidable 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 arithmetic.

## Systems of Arithmetic

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

A standard choice is Peano Arithmetic: 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

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

when  $f$  is computable.

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

## Numerals

Sine we are dealing with arbitrary arithmetic theories we have to be a bit carefule 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 arithmetic. We need to explain precisely what it means for  $T$  to express facts about some function  $f$ .

**Definition 6.** A function  $f : \mathbb{N}^n \rightarrow \mathbb{N}$  is **representable** in  $T$  if there is a formula  $\varphi$  such that

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

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

Hence, for  $f$  to be representable we need in particular

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

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

$$T \vdash \varphi(\vec{a}, a) \wedge \varphi(\vec{a}, v) \rightarrow u \approx v.$$

## Simple Examples

For example, addition and pairing are represented in any sane theory of arithmetic 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

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

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  $\Delta_0$  representation for predecessor.

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

**Exercise 6.** Show that the GCD function is representable (say, in Peano Arithmetic).

## Totality

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

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

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

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 Arithmetic. 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  $\varphi_1$  and  $\varphi_2$ , respectively.

$$\varphi(x, z) = \exists y (\varphi_1(x, y) \wedge \varphi_2(y, z))$$

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 7.** Produce a representation for arbitrary composition.

**Exercise 8.** 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  $g : \mathbb{N}^{k+1} \rightarrow \mathbb{N}$  is a function represented by  $\psi$  such that

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

Then  $f = \min g$  is represented by

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

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

## Computable is Representable

**Theorem 3.** A number-theoretic function is computable if, and only if, it is representable in Peano Arithmetic.

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

$$S(x) \neq 0 \qquad S(x) \approx S(y) \rightarrow x \approx y$$

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

$$x \neq 0 \rightarrow \exists y S(y) \approx x$$

## System $Q$

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

**Theorem 4.** System  $Q$  represents all computable functions.

This is truly amazing and somewhat tedious to prove, as one might imagine. In my regards,  $Q$  is really much too small to axiomatize arithmetic.

**Exercise 9.** Show that none of the following theorems of arithmetic 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  $\text{NP}$ .