MODEL THEORY OF ARITHMETIC

Lecture 1: The arithmetic hierarchy

Tin Lok Wong

8 October, 2014

> [These theorems] go a long way to explaining why recursion theory is relevant to the study of models of arithmetic.
>
> Richard Kaye [3, page 28]

1.1 The arithmetic hierarchy

Definition. The *language for arithmetic* $\mathscr{L}_{\mathrm{A}}$ has

- constant symbols $0, 1$;
- binary function symbols $+, \times$; and
- a binary relation symbol $<$.

The *standard model of arithmetic* is $(\mathbb{N}, 0, 1, +, \times, <)$, which we also denote by $\mathbb{N}$. All other $\mathscr{L}_{\mathrm{A}}$-structures are *nonstandard*.

Example 1.1. An $\mathscr{L}_{\mathrm{A}}$-term is just a polynomial.

The arithmetic hierarchy classifies the (quantifier) complexity of $\mathscr{L}_{\mathrm{A}}$-formulas. In this classification, quantifiers that are *bounded* are thought of as simple enough as to not contribute any complexity to the formulas in which they appear.

Definition. Assuming t is an $\mathscr{L}_{\mathrm{A}}$-term not involving the variables $\bar{x}$, we abbreviate

$$\begin{aligned} & \forall \bar{x}\ (\bar{x} < t \to \cdots) && \text{as} && \forall \bar{x}{<}t\ (\cdots) \\ \text{and} \quad & \exists \bar{x}\ (\bar{x} < t \wedge \cdots) && \text{as} && \exists \bar{x}{<}t\ (\cdots), \end{aligned}$$

where $x_1, x_2, \ldots, x_k < t$ means $\bigwedge_{i=1}^{k} x_i < t$. Such quantifiers are said to be *bounded*. A formula is Δ_0 if all its quantifiers are bounded. Let $n \in \mathbb{N}$. A Σ_n*-formula* is one of the form

$$\exists \bar{x}_1\ \forall \bar{x}_2\ \cdots\ \mathrm{Q} \bar{x}_n\ \xi(\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_n, \bar{z}),$$

where $\mathrm{Q} \in \{\forall, \exists\}$ and $\xi \in \Delta_0$. A Π_n*-formula* is one of the form

$$\forall \bar{x}_1\ \exists \bar{x}_2\ \cdots\ \mathrm{Q}' \bar{x}_n\ \zeta(\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_n, \bar{z}),$$

where $\mathrm{Q}' \in \{\forall, \exists\}$ and $\zeta \in \Delta_0$. Formulas that are equivalent to both a Σ_n- and a Π_n-formula are called Δ_n.

The definition of Δ_n is only meaningful when it is specified over which theory or in which structure the equivalence is supposed to hold.

Lemma 1.2. $\Sigma_n \cup \Pi_n \subseteq \Sigma_{n+1} \cap \Pi_{n+1}$ for all $n \in \mathbb{N}$.

Proof. The blocks of quantifiers $\bar{x}_j$ in the definition of Σ_n and Π_n can be empty. □

Every $\mathscr{L}_{\mathrm{A}}$-formula is logically equivalent to a formula in Σ_n for some $n \in \mathbb{N}$. In practice, we are mostly interested in the complexity of formulas modulo logical equivalence only.

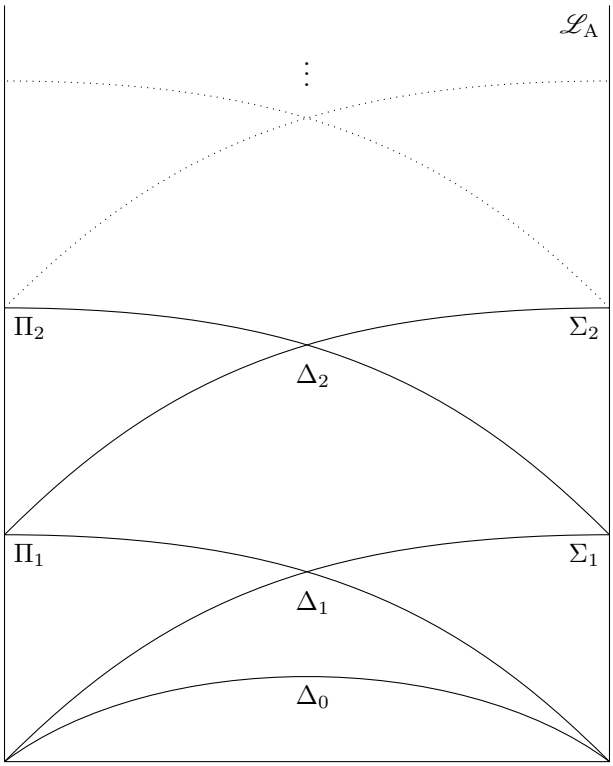

Figure 1.1: The arithmetic hierarchy

1.2 Recursion theory

Recursion theory helps one understand the arithmetic hierarchy intuitively.

Definition. Every $\mathscr{L}_{\mathrm{A}}$-formulas θ corresponds to a *program* (or *function* in the programming sense) $\mathrm{Prog}\langle\theta\rangle$ as defined recursively below. If P is a program, then $P(\bar{x}) = \delta$ means 'P on input $\bar{x}$ returns δ after finitely many steps of running'. The components for **or**, **for** and **while** below are all meant to be run *in parallel*. We interpret **if-then-else** statements in the programming way, not the logical way.

(i) If $t(\bar{x}), s(\bar{x})$ are $\mathscr{L}_{\mathrm{A}}$-terms, then $\mathrm{Prog}\langle t(\bar{x}) = s(\bar{x})\rangle$ and $\mathrm{Prog}\langle t(\bar{x}) < s(\bar{x})\rangle$ are defined by

```
boolean Prog⟨t(x̄) = s(x̄)⟩(number x̄) {
 if t(x̄) = s(x̄)
   then return true
   else return false
 end if
}

boolean Prog⟨t(x̄) < s(x̄)⟩(number x̄) {
 if t(x̄) < s(x̄)
   then return true
   else return false
 end if
}
```

(ii) If $\theta(\bar{x})$ is an $\mathscr{L}_{\mathrm{A}}$-formula, then $\mathrm{Prog}\langle\neg\theta\rangle$ is defined by

```
boolean Prog⟨¬θ⟩(number x̄) {
 if Prog⟨θ⟩(x̄) = true
   then return false
```

```
  else return true
 end if
}
```

(iii) If $\eta(\bar{x}), \theta(\bar{x})$ are $\mathscr{L}_{\mathrm{A}}$-formulas, then $\mathrm{Prog}\langle \eta \vee \theta \rangle$ is defined by

```
boolean Prog⟨η ∨ θ⟩(number x̄) {
 if Prog⟨η⟩(x̄) = true or Prog⟨θ⟩(x̄) = true
  then return true
  else return false
 end if
}
```

(iv) If $\theta(\bar{x}, y)$ is an $\mathscr{L}_{\mathrm{A}}$-formula and $t(\bar{x})$ is an $\mathscr{L}_{\mathrm{A}}$-term, then $\mathrm{Prog}\langle \exists y{<}t\ \theta \rangle$ is defined by

```
boolean Prog⟨∃y<t θ⟩(number x̄) {
 for y ← 0, 1, ..., t(x̄) − 1
  if Prog⟨θ⟩(x̄, y) = true then return true end if
 end for
 return false
}
```

(v) If $\theta(\bar{x}, y)$ is an $\mathscr{L}_{\mathrm{A}}$-formula, then $\mathrm{Prog}\langle \exists y\ \theta \rangle$ is defined by

```
boolean Prog⟨∃y θ⟩(number x̄) {
 y ← 0
 while Prog⟨θ⟩(x̄, y) = false do
  y ← y + 1
 end while
 return true
}
```

The clauses for conjunction, bounded universal quantification, and universal quantification are derived from these. Alternatively, they can be explicitly defined as follows.

(vi) If $\eta(\bar{x}), \theta(\bar{x})$ are $\mathscr{L}_{\mathrm{A}}$-formulas, then $\mathrm{Prog}\langle \eta \wedge \theta \rangle$ is defined by

```
boolean Prog⟨η ∧ θ⟩(number x̄) {
 if Prog⟨η⟩(x̄) = false or Prog⟨θ⟩(x̄) = false
  then return false
  else return true
 end if
}
```

(vii) If $\theta(\bar{x}, y)$ is an $\mathscr{L}_{\mathrm{A}}$-formula and $t(\bar{x})$ is an $\mathscr{L}_{\mathrm{A}}$-term, then $\mathrm{Prog}\langle \forall y{<}t\ \theta \rangle$ is defined by

```
boolean Prog⟨∀y<t θ⟩(number x̄) {
 for y ← 0, 1, ..., t(x̄) − 1
  if Prog⟨θ⟩(x̄, y) = false then return false end if
 end for
 return true
}
```

(viii) If $\theta(\bar{x}, y)$ is an $\mathscr{L}_{\mathrm{A}}$-formula, then $\mathrm{Prog}\langle \forall y\ \theta \rangle$ is defined by

```
boolean Prog⟨∀y θ⟩(number x̄) {
 y ← 0
 while Prog⟨θ⟩(x̄, y) = true do
  y ← y + 1
 end while
 return false
}
```

Notice programs may loop forever and return no value, because we allow **while** loops for unbounded quantification.

Remark 1.3. There is another natural way to set up the clause for $\wedge$: for $\mathscr{L}_{\mathrm{A}}$-formulas $\eta(\bar{x}), \theta(\bar{x})$, we may define

```
boolean Prog⟨η ∧ θ⟩(number x̄) {
 if Prog⟨η⟩(x̄) = true and Prog⟨θ⟩(x̄) = true
  then return true
  else return false
 end if
}
```

where the **and** returns `true` if and only if both of the conjuncts return `true`. In general, this behaves differently from what we defined, but their behaviours coincide when all the formulas involved are Δ_0.

Mental exercise 1.4. Imagine what the programs corresponding to the following formulas do.

(1) prime(x) denotes the Δ_0-formula

$$x > 1 \wedge \forall y{<}x\ \forall z{<}x\ (x \neq yz).$$

(2) not-Goldbach denotes the Σ_1-sentence

$$\exists x \left(\begin{array}{l} \exists u{<}x\ (x = 2u + 4) \\ \wedge\, \forall y{<}x\ \forall z{<}x\ \big(\mathrm{prime}(y) \wedge \mathrm{prime}(z) \rightarrow x \neq y + z\big) \end{array} \right).$$

By construction, it is apparent that whether the program $\mathrm{Prog}\langle\theta\rangle$ associated with a formula θ returns `true` on an input is closely related to whether this input satisfies θ. This is true for Σ_1-formulas, but false already for Π_1-formulas. Therefore, logically equivalent formulas may in general result in programs that halt on different inputs.

Proposition 1.5. Let $k \in \mathbb{N}$ and $\theta(x_1, x_2, \ldots, x_k) \in \Sigma_1$. Then for all $x_1, x_2, \ldots, x_k \in \mathbb{N}$,

$$\mathbb{N} \models \theta(\bar{x}) \quad \Leftrightarrow \quad \mathrm{Prog}\langle\theta\rangle(\bar{x}) = \texttt{true}.$$

Proof. First, show by induction on the complexity of θ that $\mathrm{Prog}\langle\theta\rangle$ halts on all inputs, and that the proposition is true for all $\theta \in \Delta_0$. Then one verifies the addition of unbounded existential quantifiers to Δ_0-formulas preserves the equivalence we want. □

It is believed that the programs corresponding to $\mathscr{L}_{\mathrm{A}}$-formulas capture already all concepts about $\mathbb{N}$ that can 'mechanically' be decided. Nevertheless, such a statement cannot be proved mathematically because 'mechanically' cannot be rigorously defined.

Church–Turing Thesis. For any $k \in \mathbb{N}$ and $S \subseteq \mathbb{N}^k$, the following are equivalent.

(a) There is a 'mechanical' algorithm $\mathbb{A}$ such that

$$S = \{\bar{x} \in \mathbb{N}^k : \mathbb{A} \text{ on input } \bar{x} \text{ returns } \texttt{true}\}.$$

(b) There is an $\mathscr{L}_{\mathrm{A}}$-formula θ such that

$$S = \{\bar{x} \in \mathbb{N}^k : \mathrm{Prog}\langle\theta\rangle(\bar{x}) = \texttt{true}\}.$$

Definition. Let $k \in \mathbb{N}$. A set $S \subseteq \mathbb{N}^k$ is *recursively enumerable*, or *r.e.* for short, if there is an $\mathscr{L}_{\mathrm{A}}$-formula $\theta(\bar{x})$ such that

$$S = \{\bar{x} \in \mathbb{N}^k : \mathrm{Prog}\langle\theta\rangle(\bar{x}) = \texttt{true}\}.$$

The set $S \subseteq \mathbb{N}^k$ is *recursive* if both S and $\mathbb{N}^k \setminus S$ are r.e.

Surprisingly, restricting the complexity of the formula to Σ_1 does *not* decrease the power of our programs at all.

Fact 1.6. Fix $k \in \mathbb{N}$. For every r.e. $S \subseteq \mathbb{N}^k$, there exists $\theta \in \Sigma_1$ such that

$$S = \{\bar{x} \in \mathbb{N}^k : \mathrm{Prog}\langle\theta\rangle(\bar{x}) = \texttt{true}\}.$$

Proof. Outside the scope of this course. We will merely include a sketch of the proof here; see Section 3.1 in Kaye [3] for an analogous argument in full detail.

Fix a program P such that $S = \{\bar{x} \in \mathbb{N}^k : P(\bar{x}) = \texttt{true}\}$. Write a program U_0 that halts on every given input $\bar{x}, s \in \mathbb{N}$, and returns `true` if and only if P on input $\bar{x}$ returns `true` in at most s-many steps. This can be done without the use of **while**. So $U_0 = \mathrm{Prog}\langle\eta\rangle$ for some $\eta \in \Delta_0$. For such η, we have by Proposition 1.5

$$S = \{\bar{x} \in \mathbb{N}^k : P(\bar{x}) = \texttt{true}\} = \bigcup_{s\in\mathbb{N}} \{\bar{x} \in \mathbb{N}^k : U_0(\bar{x}, s) = \texttt{true}\} = \{\bar{x} \in \mathbb{N}^k : \mathbb{N} \models \exists s\ \eta(\bar{x}, s)\}. \quad \square$$

This, together with Proposition 1.5, yields a recursion-theoretic characterization of the arithmetic hierarchy.

Corollary 1.7. (a) The r.e. sets are precisely the Σ_1-definable sets in $\mathbb{N}$.

(b) The recursive sets are precisely the Δ_1-definable sets in $\mathbb{N}$. $\square$

There are similar characterizations of formula classes higher in the arithmetic hierarchy in terms of universal machines and oracle machines. We will touch on these later in the course.

1.3 Collection

Recall from the definition of the arithmetic hierarchy that bounded quantifiers are considered immaterial when counting the complexity of an $\mathscr{L}_{\mathrm{A}}$-formula. So the levels of the arithmetic hierarchy should be invariant under bounded quantification. The axioms responsible for this invariance are known as the *collection axioms*.

Definition. Let $n \in \mathbb{N}$. The *collection scheme* for Σ_n-formulas, denoted $\mathrm{Coll}(\Sigma_n)$, is axiomatized by all sentences of the form

$$\forall \bar{z}\ \forall a\ \big(\forall \bar{x}{<}a\ \exists \bar{y}\ \theta(\bar{x}, \bar{y}, \bar{z}) \to \exists b\ \forall \bar{x}{<}a\ \exists \bar{y}{<}b\ \theta(\bar{x}, \bar{y}, \bar{z})\big),$$

where $\theta \in \Sigma_n$.

Roughly speaking, the scheme $\mathrm{Coll}(\Sigma_n)$ says that whenever we have Σ_n-definable function that has a bounded domain, the range must be bounded too. Notice the converses to collection axioms are tautologies.

Example 1.8. The standard model $\mathbb{N} \models \mathrm{Coll}(\Sigma_n)$ for every $n \in \mathbb{N}$. More generally, every regular cardinal, when viewed as an $\mathscr{L}_{\mathrm{A}}$-structure, satisfies $\mathrm{Coll}(\Sigma_n)$ for every $n \in \mathbb{N}$. Therefore, collection is, in a sense, only a cardinality property.

Theorem 1.9. Let $n \in \mathbb{N}$. Then the Σ_n- and the Π_n-formulas are both closed under bounded quantification over $\mathrm{Coll}(\Sigma_n)$.

Proof. We prove this by induction on n. The base case is clear because $\Sigma_0 = \Pi_0 = \Delta_0$.

Suppose the theorem is true for Σ_n and Π_n. Consider the Σ_{n+1}-formula $\exists \bar{y}\ \theta(\bar{x}, \bar{y}, \bar{z})$, where $\theta \in \Pi_n$. Appending a bounded existential quantifier in front does not change the complexity of this formula because existential quantifiers commute with each other, so that the induction hypothesis can easily be applied. So let us turn to bounded universal quantifiers. Given any $\mathscr{L}_{\mathrm{A}}$-term t, the formula $\forall x{<}t\ \exists \bar{y}\ \theta(x, \bar{y}, \bar{z})$ is uniformly equivalent over $\mathrm{Coll}(\Sigma_{n+1})$ to

$$\exists s\ \forall x{<}t\ \exists \bar{y}{<}s\ \theta(x, \bar{y}, \bar{z}),$$

which is equivalent to a Σ_{n+1}-formula over $\mathrm{Coll}(\Sigma_n)$ by the induction hypothesis. At the same time, $\exists x{<}t\ \forall \bar{y}\ \neg\theta(x, \bar{y}, \bar{z})$ must be equivalent to a Π_{n+1}-formula over $\mathrm{Coll}(\Sigma_{n+1})$. So we are done. $\square$

Further exercises

The appearance of the collection axioms in Theorem 1.9 is not an accident. As shown by Adamowicz and Kossak [1], under mild conditions, it is *necessary* for the collection axioms to hold if we were to have an arithmetic hierarchy that is invariant under bounded quantification.

Let $n \in \mathbb{N}$. We denote by Σ^G_{n+1} the closure of Σ_{n+1} under bounded quantification and existential quantification. Work in a fixed $\mathscr{L}_{\mathrm{A}}$-structure M in which $<$ is a linear order with no maximum element. Suppose $a, \bar{c} \in M$ and $\theta \in \Sigma_{n+1}$ such that

$$M \models \forall \bar{x}{<}a\ \exists y\ \theta(\bar{x}, y, \bar{c}) \wedge \forall b\ \exists \bar{x}{<}a\ \forall y{<}b\ \neg\theta(\bar{x}, y, \bar{c}).$$

They witness a failure of $\mathrm{Coll}(\Sigma_{n+1})$ in M.

(a) Let $\psi(u, \bar{x}) \in \Sigma^G_{n+1}$. By considering

$$\forall \bar{x}{<}a\ \exists y\ \forall u{<}y\ \big(\theta(\bar{x}, y, \bar{c}) \wedge \psi(u, \bar{x})\big)$$

or otherwise, show that $\{\bar{x} \in M : M \models \forall u\ \psi(u, \bar{x})\}$ is Σ^G_{n+1}-definable in M.

(b) Conclude that if Σ_{n+1} is closed under bounded quantification in M, then the arithmetic hierarchy *collapses* to Σ_{n+1} in M, i.e., every definable set is Σ_{n+1}-definable in M.

In this course, we will see virtually no $\mathscr{L}_{\mathrm{A}}$-structure in which the arithmetic hierarchy collapses.

Further reading

There is a very good reason why we can usually ignore bounded quantifiers. Recall a formula is *existential* if it contains no universal quantifiers.

MRDP Theorem. For every Σ_1-formula $\theta(\bar{x})$, there exists an existential $\mathscr{L}_{\mathrm{A}}$-formula $\theta'(\bar{x})$ such that $\mathbb{N} \models \forall \bar{x}\ \big(\theta(\bar{x}) \leftrightarrow \theta'(\bar{x})\big)$.

We can read this as saying that over the standard model, bounded quantifiers can be eliminated without affecting quantifier complexity. It is key to the solution of Hilbert's Tenth Problem. Both Matiyasevich's book [4] and Davis's book [2] contain an exposition and a proof of this theorem. The other two letters in the name of the theorem refer to Julia Robinson and Hilary Putnam.

References

[1] Zofia Adamowicz and Roman Kossak. A note on $\mathrm{B}\Sigma_n$ and an intermediate induction schema. *Mathematical Logic Quarterly*, 34(3):261–264, 1988.

[2] Martin Davis. *Computability and Unsolvability*. Dover Publications, Mineola, 1982. Reprint of the McGraw-Hill, New York, 1958 edition. Includes a new appendix "Hilbert's Tenth Problem is Unsolvable" originally appeared in the *American Mathematical Monthly*.

[3] Richard Kaye. *Models of Peano Arithmetic*, volume 15 of *Oxford Logic Guides*. Clarendon Press, Oxford, 1991.

[4] Yuri V. Matiyasevich. *Hilbert's Tenth Problem*. Foundations of Computing Series. The MIT Press, Cambridge, 1993. With a foreword by Martin Davis.

[5] Charles Parsons. On a number theoretic choice schema and its relation to induction. In Akiko Kino, John Myhill, and Richard E. Vesley, editors. *Intuitionism and Proof Theory*, volume 60 of *Studies in Logic and the Foundations of Mathematics*, pages 459–473. North-Holland Publishing Company, Amsterdam, 1970.