# COMP9103 - 07s2
# Complexity and Logic

Aleksandar Ignjatović

July 30, 2007 – Week 1

## 1 Introduction

We consider classes of functions of various computational complexity, such as

$$\ldots \subset \text{P-Time} \subset \text{Primitive Recursive} \subset \text{Recursive} \subset \ldots$$

Recursive functions: computable on a TM; might be partial (i.e., the TM might not stop for some inputs). Intuitively, recursive functions are computable using `WHILE`-loop – without any priori bound on the number of iterations of the loop. (*Cook the potatoes until tender.*)

Primitive recursive functions: A smaller class, computable using a loop of the form `FOR` $i < g(x)$ `DO` $\cdots$, where $g(x)$ is previously defined primitive recursive function. (*Cook the potatoes for ten minutes.*)

More formally:

**Definition 1.**

- $f(x, \vec{y})$ *is defined by* primitive recursion *from* $g(\vec{y})$, $h(x, \vec{y}, z)$ *if*

$$f(0, \vec{y}) = g(\vec{y})$$
$$f(x+1, \vec{y}) = h(x, \vec{y}, f(x, \vec{y}))$$

- $f(\vec{x})$ *is obtained by* composition *from* $h(y_1, y_2, \ldots, y_n)$ *and* $g_1(\vec{x})$, $g_2(\vec{x})$, $\ldots$, $g_n(\vec{x})$ *if*

$$f(\vec{x}) = h\left(g_1(\vec{x}), g_2(\vec{x}), \ldots, g_n(\vec{x})\right)$$

*Primitive recursive functions* are the smallest class containing functions $\mathtt{o}(x) = 0$, $\pi_i^n(x_1, \ldots, x_n) = x_i$ and $S(x) = x + 1$, and which is closed for composition and primitive recursion.

Examples:

$$x + 0 = x \tag{1}$$
$$x + (y + 1) = (x + y) + 1 \tag{2}$$

can be seen as definition of $x+y$ by primitive recursion from $g(x)=\pi_1^1(x)$ and $h(x)=S(x)$, or, with the usual prefix functional notation,

$$\begin{aligned} \oplus(x,0) &= \pi_1^1(x) && (3)\\ \oplus(x,S(y)) &= S(\oplus(x,y)) && (4)\end{aligned}$$

Similarly, we can define $x\cdot y$ by primitive recursion from $\mathfrak{o}(x)$ and $+$:

$$\begin{aligned} x\cdot 0 &= 0 \quad (=\mathfrak{o}(x))\\ x\cdot(y+1) &= (x\cdot y)+x\end{aligned}$$

and $x^y$ from $\mathfrak{o}(x)$, $S(x)$ and multiplication:

$$\begin{aligned} x^0 &= 1 \quad (=S(\mathfrak{o}(x)))\\ x^{y+1} &= x^y\cdot x\end{aligned}$$

If we set $f_0(x,y)=S(x)$, $f_1(x,y)=x+y$, $f_2(x,y)=x\cdot y$, $f_3(x,y)=x^y$, we have:

$$\begin{cases} f_1(x,0) &= x\\ f_1(x,y+1) &= f_0(f_1(x,y),x)\end{cases}$$
$$\begin{cases} f_2(x,0) &= 0\\ f_2(x,y+1) &= f_1(f_2(x,y),x)\end{cases}$$
$$\begin{cases} f_3(x,0) &= 1\\ f_3(x,y+1) &= f_2(f_3(x,y),x)\end{cases}$$

Thus, $f_4(x,y)$ can be defined as:

$$\begin{cases} f_4(x,0) &= x\\ f_4(x,y+1) &= f_3(f_4(x,y),x)\end{cases}$$

i.e., $f_4(x,y)=\underbrace{x^{\cdot^{\cdot^{\cdot^{x}}}}}_{y \text{ copies}}$. In general, we can set

$$\begin{cases} f_{n+1}(x,0) &= g_n(x)\\ f_{n+1}(x,y+1) &= f_n(f_{n+1}(x,y),x).\end{cases}$$

and we have

$$\begin{aligned} f_{n+1}(x,y) &= f_n(f_{n+1}(x,y-1),x)\\ &= f_n(f_n(f_{n+1}(x,y-2)),x)\\ &= \underbrace{f_n(f_n(\ldots,f_n}_{y \text{ copies}}(g_n(x),x),\ldots,x)\end{aligned}.$$

Considering $n$ as just a variable, we have:

$$\begin{aligned} F(n+1,x,0) &= g_n(x)\\ F(n+1,x,y+1) &= F(n,F(n+1,x,y),y)\end{aligned}$$

Thus, $F$ is obtained by a <u>double recursion</u>. Slightly simplifying this definition we get

**Definition 2** (The Ackermann function)**.**

$$\begin{aligned} f(0,y) &= y+1 \\ f(x+1,0) &= f(x,1) \\ f(x+1,y+1) &= f(x,f(x+1,y)) \end{aligned} \tag{5}$$

Why is this a correct definition of a function $\mathbb{N} \mapsto \mathbb{N}$? We can define $f$ by setting

$$f(m,n) = k \leftrightarrow \text{there exists a verifying computation } \mathcal{C}(m,n,k).$$

Here $\mathcal{C}(m,n,k) = \{eq(1),\ldots,eq(p)\}$ is a set of equations such that $eq(p) \equiv f(m,n) = k$ and every $eq(i)$, $1 \leqslant i \leqslant p$, is an instance of the defining axioms (by composition or primitive recursion) for the functions involved or obtained from some previous equations using some closed (i.e., variable free) instances of the axioms of equality:

$$\begin{aligned} Ax =: \quad & x = x, \\ & x = y \to y = x, \\ & (x = y) \wedge (y = z) \to x = z, \\ & (t_1 = \bar{t}_1) \wedge \ldots \wedge (t_k = \bar{t}_k) \to g(t_1,\ldots,t_k) = g(\bar{t}_1,\ldots,\bar{t}_k) \\ & \text{(for all functions } g \text{ and all terms } t_i \text{ involved in the computation)} \end{aligned}$$

Example:

$\{f(1,1) = f(0{+}1,0{+}1); f(0{+}1,0{+}1) = f(0,f(1,0)); f(1,0) = f(0,1); f(0,1) = 1+1; f(0,1) = 2; f(1,1) = f(0,2); f(0,2) = 2+1; f(1,1) = 3\}$ verifies that $f(1,1) = 3$.

Thus, $f(m,n) = k \leftrightarrow \exists \mathcal{C}(\text{“}\mathcal{C} \text{ is a computation of } f(m,n) = k\text{”})$.

Assume now that $f$ is not defined correctly by (5), i.e., that for some $m$, $n$, there is no $\mathcal{C}$ as above. Let $m$ be the least number such that there exists $n$ such that for no $\mathcal{C}, k$ the above holds, i.e., the least $m$ such that

$$\exists n \forall \mathcal{C}, k(\text{“}\mathcal{C} \text{ is } \textbf{not} \text{ a computation of } f(m,n) = k\text{”}). \tag{6}$$

Take now the least such $n$ to obtain a contradiction. Similarly, one can show that there exists at most one $k$ such that $f(m,n) = k$.

**Definition 3.**

1. $\Sigma_0 = \Pi_0$ *is the smallest set of formulas that contains the atomic formulas* $t_1 = t_2$ *and* $t_1 < t_2$ *and the negations of atomic formulas, and which is closed for conjunctions, disjunctions and bounded quantifiers* $\forall x < t(\vec{y})$ *and* $\exists x < t(\vec{y})$.

2. $\Sigma_n$ *is the smallest set that contains* $\Pi_{n-1}$ *formulas and which is closed for conjunctions, disjunctions,* $\exists x$ *and* $\forall x < t(\vec{y})$.

3. $\Pi_n$ *is the smallest set that contains* $\Sigma_{n-1}$ *formulas and which is closed for conjunctions, disjunctions,* $\forall x$ *and* $\exists x < t(\vec{y})$.

Question: The above formula (6) is $\Sigma_2$. Can we prove that $f$ is well defined using the least number principle for $\Sigma_1$ formulas only?

Answer: We will see that the answer is **no**. This hints to the first link between computational complexity and logic: inherent complexity of an algorithm defining a function $f(\vec{x})$ can be "measured" by the strength of the "weakest" theory in which one can prove that the algorithm is well defined, i.e., such that

$$\mathbf{T} \vdash \forall \vec{x}\ \exists! y(\text{“}f \text{ on input } \vec{x} \text{ converges to } y\text{”})$$

and such that for every other theory $\mathbf{T}'$ (from the same class) that proves the same statement, we have $\mathbf{T} \subseteq \mathbf{T}'$.

**Theorem 1.** *Ackermann function is* **not** *primitive recursive. Thus, the double recursion cannot be replaced by several instances of single recursion and composition. Note, however, that for every fixed natural number $n$, function $f(n, y)$ of the single variable $y$* **is** *primitive recursive!*

The proof is based on several lemmas.

**Lemma 2.**

1. *function $f(x, y)$ is monotonic in both arguments, i.e., $f(x+1, y) > f(x, y)$, $f(x, y+1) > f(x, y)$; thus, also $f(x, y) > y$ and $f(x, y) > x$;*
2. *function $f(x, y)$ grows faster in $x$ than in $y$, i.e., $f(x+1, y) > f(x, y+1)$.*

*Proof.* Exercise 1; use double induction (inductive step on $x$ is proved by induction on $y$, for all inequalities simultaneously).

Note: Exercises are homework due in one week's time. □

**Lemma 3.** $f(1, y) = y + 2, f(2, y) = 2y + 3$

*Proof.* Exercise 2. □

**Lemma 4.** *For all $c_1, c_2, \ldots, c_n$, there exists $c$ such that*

$$\sum_{j=1}^{n} f(c_j, x) < f(c, x)$$

*Proof.* Enough to prove for $n = 2$, left as Exercise 3. Hint: let $c = max(c_1, c_2)$; then $f(c_1, x) + f(c_2, x) < 2f(c, x) < 2f(c, x) + 3 = f(2, f(c, x)) < f(c+4, x)$ □

**Lemma 5** (The Main Lemma)**.** *For every primitive recursive function $h(\vec{x})$, there exists $c$ such that*

$$h(x_1, \ldots, x_n) < f(c, x_1 + \cdots + x_n)$$

*Proof.* Exercise 4. Hint: use induction on the complexity of definition of $h(\vec{x})$. Show that lemma holds for $S(x) = x + 1$, $\mathfrak{o}(x) = 0$ and $\pi_i^n(x_1, \ldots x_n) = x_i$. Then show that the property is preserved for functions defined by composition

and primitive recursion, using the previous lemmas. The key step is induction on $y$; show that if

$$\begin{aligned} h(y+1,\vec{x}) &= g(y,\vec{x},h(y,\vec{x})),\\ g(y,\vec{x},z) &< f(c_1,\sum\vec{x}+y+z),\\ h(y,\vec{x}) &< f(c,\sum\vec{x},y) \quad \text{for some } c>c_1, \end{aligned}$$

then also

$$h(y+1,\vec{x}) < f(c,\sum\vec{x}+y+1)$$

□

Proof of the Theorem (1)

Assume $f(x,y)$ were primitive recursive, then so would be $f(x,x)$ and thus for some $c$ we would have $f(x,x) < f(c,x)$; take $x=c$ and get $f(c,c) < f(c,c)$. Contradiction!

Thus, there are recursive (TM computable) functions that are not primitive recursive.

We saw that one can show that the Ackermann function is well defined using the least number principle for $\Sigma_2$ formulas. So perhaps if we restrict ourselves to the least number principle for $\Sigma_1$ formulas, we can characterize the primitive recursive functions as functions whose definition can be shown to be correct using the least number principle for $\Sigma_1$ formulas only:

$$\exists x\exists y\varphi(x,y) \to \exists x(\exists y\varphi(x,y) \wedge (\forall z<x)\neg\exists y\varphi(z,y))$$

where $\varphi$ contains only bounded quantifiers. For this purpose we must introduce formal theories of arithmetic.

# 2 Formal Theories of Arithmetic

Language: $L_{PA} = \{+,\cdot,0,S,<\}$ or richer.

Robinson's Q: The domain of numbers is linearly ordered, with 0 as the least element:

$$\begin{aligned} &(x<y)\wedge(y<z)\to x<z\\ &(x<y)\to\neg(y<x)\\ &(x<y)\vee(x=y)\vee(x>y)\\ &(x>0)\vee(x=0) \end{aligned}$$

The successor function is injective and 0 is not in its range:

$$\begin{aligned} &S(x)=S(y)\to x=y\\ &\neg(S(x)=0) \end{aligned}$$

The ordering is discrete:

$$x<S(y)\to(x<y)\vee(x=y)$$

The primitive recursive definitions of addition and multiplication:

$$\begin{array}{ll} x+0=x & x\cdot 0=0 \\ x+S(y)=S(x+y) & x\cdot S(y)=x\cdot y+x \end{array}$$

Also, $\mathrm{I}\Sigma_1 = \mathrm{Q} \cup \{\varphi(0,\vec{y}) \wedge \forall x(\varphi(x,\vec{t}) \to \varphi(x+1,\vec{y})) \to \forall x\varphi(x,\vec{y}) \ : \ \varphi \in \Sigma_1\}$

The *Peano Arithmetic* is obtained by adding the full induction schema, for formulas of arbitrary complexity.

**Definition 4.** *A function $f : \mathbb{N}^n \mapsto \mathbb{N}$ is provably total in a theory $\mathbf{T}$ iff there exists a formula $\varphi$ of an appropriate complexity such that $\mathbf{T} \vdash \forall\vec{x}\exists! y\varphi(\vec{x},y)$ and $\mathbb{N} \models \forall\vec{x}\varphi(\vec{x},f(\vec{x}))$. Here "appropriate complexity" depends on the power of $\mathbf{T}$.*

**Theorem 6** (Mints, Parsons)**.** *Provably total functions in $\mathrm{I}\Sigma_1$ with a $\Sigma_1$ graph $\varphi(\vec{x},y)$ are precisely the primitive recursive functions, i.e.,*

$$\mathrm{I}\Sigma_1 \vdash \forall\vec{x}\exists! y\varphi(\vec{x},y) \tag{7}$$

*for some $\varphi \in \Sigma_1$ just in case*

$$\mathbb{N} \models \forall\vec{x}\varphi(\vec{x},f(\vec{x})) \tag{8}$$

*for some primitive recursive function $f(\vec{x})$.*

We will present two proofs of the above theorem, one proof-theoretic and one model-theoretic, because both are typical of their kind.

We first show the "easy" direction – i.e., we show that for every primitive recursive function $f(\vec{x})$ there exists a $\Sigma_1$ formula $\varphi(\vec{x},y)$ such that (7), (8) hold.

The key ingredient is coding of sequences of arbitrary length, finite from model's point of view but *possibly infinite* from the "outside" the model.

To understand this, we must look at nonstandard models of arithmetic. A model of arithmetic is nonstandard if it is not (isomorphic to) $\mathbb{N}$.

Consider the theory $\mathbf{T} = Th(\mathbb{N}) \cup \{c > \underline{n} \,|\, n \in \mathbb{N}\}$ where

$$\underline{n} = \underbrace{S(S\ldots S}_{n \text{ copies}}(0)\ldots)$$

Every finite subset $\mathbf{S} \subset \mathbf{T}$ is consistent because it contains only finitely many inequalities of the form $c > \underline{n}$. Thus, one can choose $m$ larger than any $n$ such that $c > \underline{n}$ belongs to $\mathbf{S}$ and interpret $c$ as $\underline{m}$. Thus, by compactness, $\mathbf{T}$ is consistent and thus by completeness it has a model $\mathfrak{M}$. Clearly, in such a model, $\mathfrak{M} \models c > \underline{n}$ for every $n$, i.e., such model is nonstandard.

But what does such model look like? Completeness or compactness offer little clue, except for a few general features.

Exercise: Prove that the order type of every countable non-standard model of $\mathrm{Q}$ is $\mathbb{N} + \mathbb{Q}\cdot\mathbb{Z}$ where $\mathbb{Q}$ is the order type of the rational numbers (dense linear ordering without end points) and $\mathbb{Z}$ is the order type of integers.

So perhaps Completeness is not the best way to obtain non-standard models of arithmetic, because the Henkin construction is not informative about the structure of a model obtained from the Henkin constants.

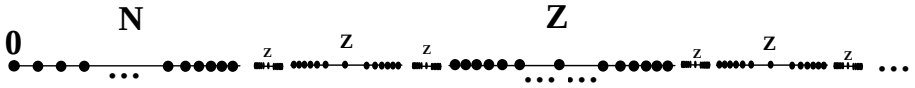


Figure 1: A nonstandard model: $\mathbb{N}$ followed by a dense set of copies of $\mathbb{Z}$.

Let's look at the filter on $\mathbb{N}$ generated by the co-finite sets (sets of natural numbers whose complements are finite).

What is a filter on a set $S$? A collection $\mathfrak{F}$ of subsets of $X$ such that $\forall X, Y \in \mathfrak{F}(X \cap Y \in \mathfrak{F})$ and $\forall X, Y\big(X \in \mathfrak{F} \wedge X \subset Y \to Y \in \mathfrak{F}\big)$.

An ultrafilter is a filter if it is not contained as a proper sub-collection of any other filter, i.e., it is a maximal filter (in the sense of inclusion).

Claim: If $\mathfrak{F}$ is an ultrafilter on $X$ then for every $A \subset X$ either $A$ or $(X \setminus A) \in \mathfrak{F}$.

*Proof.* Otherwise, if $A \notin \mathfrak{F}$, since $\mathfrak{F}$ is an ultrafilter, $A$ could not be added to $\mathfrak{F}$, and thus there would be a collection $X_1, \ldots, X_n \in \mathfrak{F}$ s.t. $A \cap X_1 \cap \ldots \cap X_n = \emptyset$ and similarly also $Y_1, \ldots, Y_m \in \mathfrak{F}$ s.t. $(X \setminus A) \cap Y_1 \cap \ldots \cap Y_m = \emptyset$. i.e., $X_1 \cap \ldots \cap X_n \cap Y_1 \cap \ldots \cap Y_m = \emptyset$. Contradiction! □

Let now $M = \prod_{i \in \mathbb{N}} \mathbb{N}$; thus, elements of $M$ are functions $\mathbb{N} \to \mathbb{N}$ and define for $\vec{x}, \vec{y} \in \mathbb{N}^{\mathbb{N}}$

$$\vec{x} \equiv \vec{y} \leftrightarrow \{i \,|\, x(i) = y(i)\} \in \mathfrak{F};$$
$$\vec{x} \leqslant \vec{y} \leftrightarrow \{i \,|\, x(i) \leqslant y(i)\} \in \mathfrak{F}$$

The corresponding quotient structure $A/\equiv$ consisting of equivalence classes of $\equiv$ is denoted $\prod_{\mathfrak{F}} \mathbb{N}$, with operations defined coordinate-wise.

Claim: $\prod_{\mathfrak{F}} \mathbb{N} \models \varphi$ just in case $\mathbb{N} \models \varphi$.

In general if $\mathfrak{M} \models \prod_{i \in \mathfrak{F}} \mathfrak{M}_i$ then $\mathfrak{M} \models \varphi$ just in case $\{i \,|\, \mathfrak{M}_i \models \varphi\} \in \mathfrak{F}$.

*Proof.* An exercise. □

Claim: $\prod_{i \in \mathfrak{F}} \mathbb{N}$ is non-standard.

*Proof.* Let $[f]$ denote the equivalence class of an element $f \in \prod_{i \in \mathbb{N}} \mathbb{N}$; Clearly, for $f(i) = i$, $[f] > [\underline{n}]$ for all $n$. □

But is this construction more informative than Henkin's? We have no control over what sentences $\mathfrak{F}$ makes true more than what completion in Henkin's construction does.

Can we construct an "understandable" non-standard model of arithmetic??

In a sense, no, if such theory of arithmetic has even very modest power, as shown by Stanley Tennenbaum. In fact, Henkin's construction can be made optimal just by being careful how we choose the completion.

**Theorem 7.** *(Tennenbaum [1]) Let $\mathfrak{M} \models Th(\mathbb{N})$ and let $\mathfrak{M}$ be countable and non standard. Then $+^{\mathfrak{M}}$ and $\cdot^{\mathfrak{M}}$ are not recursive.*

*Proof.* We will show later that for every $\varphi(x)$ there exists a nonstandard element $c \in M$ such that

$$\mathfrak{M} \models \varphi(\underline{n}) \text{ just in case } \mathfrak{M} \models \underline{P_n} \,|\, c$$

where $P_n$ is the $n^{th}$ prime and $P_n \,|\, c$ means $P_n$ divides $c$.

Thus, since $\mathfrak{M} \models Th(\mathbb{N})$ also $\mathbb{N} \models \varphi(\underline{n})$ just in case $\mathfrak{M} \models \exists c(c +^{\mathfrak{M}} \ldots +^{\mathfrak{M}} c = c_1)$ and $\mathbb{N} = \neg\varphi(\underline{n})$ just in case $\mathfrak{M} \models \exists c(c +^{\mathfrak{M}} \ldots +^{\mathfrak{M}} = c_2)$, where $c_1$ and $c_2$ are two particular fixed elements of $\mathfrak{M}$. Thus, every arithmetic set is recursive in $+^{\mathfrak{M}}$.

Consequently, $+^{\mathfrak{M}}$ cannot be recursive because otherwise the Arithmetic Hierarchy would collapse, and this is not the case, as we will see. Similar proof works for $\cdot^{\mathfrak{M}}$.

Note that here the existence of non-standard elements in $\mathfrak{M}$ is used to collapse the complexity of definition of arithmetic sets. □

[1] Originally more general then what we prove below, just to give an idea.