CS 599: The Meta-Complexity Frontier, Fall 2023
Problem Set #3

Due: 5:00PM, Friday, December 21, 2023.

Homework Policies

- Submit your completed assignment by email to `marco[at]ntime[dot]org` . Please include the string "CS599PS2" somewhere in your subject line.
- Solutions must be typeset, e.g., using LaTeX or Microsoft Word.
- You are encouraged to collaborate on the homework problems with each other in small groups (2 - 3 people). Collaboration may include brainstorming or exploring possible solutions together on a whiteboard, but should not include one person telling the others how to solve a problem. You **must write up the solutions independently** (in your own words) and **acknowledge your collaborators** at the beginning of the first page.
- You may read papers and other outside sources to help you solve these problems. If you do so, you **must cite and acknowledge any sources and write the solutions in your own words.**
- You may freely use without proof any results proved in class, in lecture notes posted on the class webpage, or in the main body of the texts assigned as reading. Note that this excludes results that appear in the texts as problems and exercises. You may, of course, use such results but you have to prove them first.
- To help your instructor calibrate the length and difficulty of future assignments, please include with each problem an estimate of how long it took you to solve it.
- Please start early! The problems are presented roughly in the order of the course content they correspond to, so you may get started on the first few problems as soon as the assignment is released. Late assignments will receive credit only with prior permission of the instructor.

Part of this and subsequent assignments will be to familiarize yourself with definitions of complexity classes and concepts that did not come up in class. These items will always be defined in our "local" complexity zoo, linked from the `Resources` section of our course webpage.

1 Warm-up Inside a Theory of Bounded Arithmetic

Hear you will prove a few basic facts about arithmetic "inside" a particularly weak fragment of Peano Arithmetic. We'll specify the theory $I\Delta_0$ which is **single-sorted** and thus more succinct than what we have seen in class so far. First, some basic definitions from logic:

1. The *vocabulary of arithmetic* is $\mathcal{L}_A = \{0, 1, +, \cdot, =, \leq\}$ where 0 and 1 are constant symbols, $+, \cdot$ are binary function symbols, and $=, \leq$ are binary relation symbols.
2. The $\mathcal{L}_A$*-formulas* are built by induction on syntax from symbols in $\mathcal{L}_A$ with the logical connectives $\{\wedge, \vee, \neg, \rightarrow\}$, quantifiers $\{\forall, \exists\}$, an infinite set of variables $x_1, x_2, \ldots$, and parentheses (see Item 0.3).
3. An $\mathcal{L}_A$-structure M is a set U (the universe) together with *interpretations* of all the symbols:

- For the constants 0 and 1, distinguished elements $0^M, 1^M \in U$
- For the relation $\leq$, a set of pairs $R_{\leq} \in U \times U$
- For the functions $+$ and $\cdot$, mappings $+^M, \cdot^M : U \times U \to U$.

Notice that these interpretations may have nothing whatsoever to do with the standard meaning we ascribe to the symbols of $\mathcal{L}_A$; these structures are merely *suitable* for evaluation of $\mathcal{L}_A$-formulas. The only constraint is that $=$ is always interpreted as the actual identity relation on U (see Item 0.4).

4. An $\mathcal{L}_A$-structure M *satisfies* a formula φ — denoted $M \models \varphi$ — if (intuitively) "φ is true in M." Satisfaction can be defined formally using induction on the parse tree of φ (see Item 0.6) or a two-player game (see Section 2). A model M satisfies a set of formulas Γ if $M \models \varphi$ for every $\varphi \in \Gamma$. We also say "M *is a model* of φ" when M satisfies φ.

5. Let Γ be a set of formulas, and let φ be a formula. If every model of Γ is also a model of φ, we say that φ is a *logical consequence* of Γ — denoted (unfortunately) $\Gamma \models \varphi$.

6. Finally, a *theory* is (semantically) a set of formulas $\mathcal{T}$ closed under logical consequence. We could also define a theory syntactically by closing a set of formulas under deduction (see Item 0.11).

Now consider the following set of formulas, our *basic axioms*:

B1. $(\forall x)\ x + 1 \neq 0$

B2. $(\forall x)(\forall y)\ [x + 1 = y + 1] \to x = y$

B3. $(\forall x)\ x + 0 = x$

B4. $(\forall x)(\forall y)\ x + (y + 1) = (x + y) + 1$

B5. $(\forall x)\ x \cdot 0 = 0$

B6. $(\forall x)(\forall y)\ x \cdot (y + 1) = (x \cdot y) + x$

B7. $(\forall x)(\forall y)\ [x \leq y \wedge y \leq y] \to x = y$

B8. $(\forall x)(\forall y)\ x \leq x + y$

We could define a theory using axioms B1-8 alone, but it would be extremely weak — able to prove statements like $(\forall x)(\forall y)\ x + y = 0 \to x = 0 \wedge y = 0$, but not much more. To strengthen the theory, we consider adding an infinite collection of formulas — a *scheme* — to the axioms.

Definition 1 (Induction Scheme)**.** For Φ a set of formulas the Φ-Ind axiom scheme is the set of formulas

$$[\varphi(0) \wedge (\forall x)\ [\varphi(x) \to \varphi(x + 1)]] \to (\forall z)\ \varphi(x)$$

where $\varphi \in \Phi$.

The theory of Peano Arithmetic allows induction over *any* formula of $\mathcal{L}_A$, and so is too powerful for our computationally-limited purposes. To build theories related to complexity classes, we restrict the formulas Φ over which induction is allowed so that proofs cannot construct objects that are "too big."

Definition 2 (Bounded Quantifiers)**.** If the variable x does not occur in the term t, then $\exists x \leq t\varphi$ stands for $\exists x(x \leq t \wedge \varphi)$ and $\forall x \leq t\varphi$ stands for $\forall x(x \leq t \to \varphi)$. Quantifiers that occur in this form are called *bounded* and a *bounded formula* is one in which every quantifier is bounded.

We can now define our theory: powerful enough to be interesting, weak enough to use as a running example in this problem set and illustrate some of the limitations of bounded arithmetic and motivation for stronger theories.

Definition 3 (Theory $I\Delta_0$)**.** Denote by Δ_0 the set of bounded formulas over $\mathcal{L}_A$. The theory $I\Delta_0$ has as axioms B1-B8 and Δ_0-Ind.

$I\Delta_0$ *can* prove many basic facts about number theory, *cannot prove* that $x^{\log x}$ is a total function, and *seems unable* to prove that there are arbitrarily large prime numbers, even though primality is a definable predicate (Definition 4 below) of $I\Delta_0$.

⋆**Problem:** Argue that the following are theorems of $I\Delta_0$. You may describe a proof in a Hilbert system (see Item 0.10) or a winning strategy for Elouise in an arbitrary model of $I\Delta_0$ (as in class). Be precise about how induction is used. Hint: the theorems build on each other.

1. Addition is commutative: $(\forall x)(\forall y)x + y = y + x$
2. Predecessor: $(\forall x)x \neq 0 \to \exists y \leq x(x = y + 1)$
3. Difference: $(\forall x)(\forall y)(\exists z)[x + z = y \vee y + z = x)$

2 Relating $I\Delta_0$ to Complexity Classes

First we give a theorem that quantifies just how "bounded" $I\Delta_0$ is. Informally, it cannot "map" free variables ($\forall\exists$ quantifier prefix) to a number that is "too big" compared to the inputs.

Theorem 1 (Parikh's Theorem)**.** *Suppose the formula $(\forall x_1)(\forall x_2)\dots(\forall x_k)(\exists y)\varphi(x_1,\dots,x_k,y)$ with all free variables displayed is a theorem of $I\Delta_0$. Then, there is a term t of $\mathcal{L}_A$ involving only the x variables such that $(\forall x_1)(\forall x_2)\dots(\forall x_k)(\exists y \leq t)\varphi(x_1,\dots,x_k,y)$ is also a theorem of $I\Delta_0$.*

In class, we saw a (two-sorted) language of arithmetic expanded with all polytime function symbols. When is it "legitimate" to expand a language with predicate symbols? One answer is, if we could have written the predicate already using symbols in the language — then the new symbol can be treated as an abbreviation.

Definition 4 (Definable Predicates)**.** Let $\mathcal{T}$ be a theory with vocabulary $\mathcal{L}_A$. A k-ary predicate symbol $P(x_1,\dots,x_k)$ not in $\mathcal{L}_A$ is Φ-definable in $\mathcal{T}$ if there is an $\mathcal{L}_A$-formula $\varphi(x_1,\dots,x_k)$ in Φ such that

$$P(x_1,\dots,x_k) \iff \varphi(x_1,\dots,x_k)$$

This is called the *defining axiom* for P.

The related complexity class of interest is a brittle analog of the familiar polynomial-time hierarchy.

Definition 5 (Linear Time Hierarchy)**.** Let $\mathsf{NLinTime}$ denote time $O(n)$ on a nondeterministic k-tape Turing Machine. Then $\Sigma_0^{lin} = \mathsf{NLinTime}$ and, inductively equipping previous layers of the hierarchy with an oracle,

$$\Sigma_{i+1}^{lin} = \mathsf{NLinTime}^{\Sigma_i^{lin}}$$

and

$$\mathsf{LTH} = \bigcup_{i\in\mathbb{N}} \Sigma_i^{lin}$$

This complexity class is closely related to $I\Delta_0$.

Theorem 2. LTH *is exactly the set of Δ_0-definable predicates, where the Turing Machines get binary strings encoding numbers as inputs and formulas get the corresponding numbers as assignments to their free variables.*

⋆**Problem:** Sketch a proof (outside $I\Delta_0$) of theorem 2. Here are some assumptions and hints.

- **Assume** that binary sequences can be coded in $I\Delta_0$, so that predicates like Bcount(x) = "the number of ones in the binary representation of x" are Δ_0-definable, as well as extraction of specific bits and indexing into arrays of bitstrings. Substantial "bootstrapping" is required to obtain these definitions in $I\Delta_0$ — out of scope for this class. So, just assume that any function on bitstrings in AC^0 is Δ_0-definable.
- **Hint:** the base case of one direction is Δ_0-def inability of every language in $\mathsf{NLinTime}$.
- **Hint:** be inspired by the proof of the Cook-Levin theorem.
- **Hint:** the other direction is by structural induction on Δ_0 formulas. Use Parikh's theorem.