COMPUTABILITY III
TOPICS IN COMPUTABILITY

ARISTOTELIS PANAGIOTOPOULOS

Intro

This is a topics course in computability. Among others we will cover:

- undecidability of the word problems for groups;
- quantifier elimination and decidable theories;
- computational complexity;
- dependence of P=NP from oracles;
- The Paris Harrington theorem and the independence of Ramsey theoretic statements from Peano arithmetic;
- recursion theoretic forcing and undefinability of definability.

Time permitting we will also cover some basic effective descriptive set theory.

Contents

1. Some more examples of undecidable theories

Recall our technique for showing that a theory T is undecidable: find a strongly undecidable structure $\mathcal{M}$ with $\mathcal{M} \models T$.

To show that an $\mathcal{L}$-structure $\mathcal{M}$ is strongly undecidable it suffices to show that some other strongly undecidable structure $\mathcal{M}'$ in a possibly different **finite** language $\mathcal{L}'$ is definable in $\mathcal{M}$ (perhaps with parameters).

We have already established that $\mathcal{N} = (\mathbb{N}, 0, S, +, *, <)$ is strongly undecidable and that there is a strongly undecidable graph $\mathcal{G} = (V, E)$.

Let us remind the technique by using it to show that the theory of posets and that the theory of lattices is undecidable. Recall that a **poset** is a structure $\mathcal{P} = (P, <)$ where $<$ is a binary relation so that

(1) $\mathcal{P} \models \forall x \forall y \forall z \left((x < y \wedge y < z) \implies x < z\right)$;
(2) $\mathcal{P} \models \forall x \forall y \, (x < y \implies \neg(x = y \vee y < x)$;

A poset is a **lattice** if every finite subset F of P has a unique least upper bound and a greatest lower bound.

Theorem 1. *There is a strongly undecidable lattice $\mathcal{P} = (P, <)$. As a consequence the theory of posets and the theory of lattices is undecidable.*

Proof. Let $\mathcal{G} = (V, E)$ be a strongly undecidable graph. We will define a lattice $\mathcal{P} = (P, <)$ in which $\mathcal{G}$ is definable.

Let P be the set of all subsets A of $V \cup E$ with the property that

$$(a, b) \in A \implies a \in A, b \in A$$

Let also $<$ be the inclusion, i.e., set $A < B$ if and only if $A \subsetneq B$. Notice that P is closed under $\cap, \cup$ and therefore it is a lattice. In fact, since $\emptyset, V \cup E \in P$ it is a bounded lattice.

Let $V' \subset P$ be the collection of all atoms of the lattice which is definable:

$$A \in V' \iff A \neq \emptyset \wedge \forall B (B < A \implies B = \emptyset),$$

and into one-to-one correspondence with V. Let also E' be the binary relation on V' defined by the formula

$$AE'B \iff \exists C \exists D \Big((A, B < C < D) \wedge$$

$$\forall D' (D' < D \wedge D' \in V' \implies D' = A \vee D' = B) \Big).$$

It is easy now to see that the assignment $(V, E) \mapsto (V', E')$ provides a definition of $\mathcal{G}$ inside $\mathcal{P}$. □

A language $\mathcal{L}$ is called **non-trivial** if $\mathcal{L}$ contains a binary relation or a binary function symbol or at least two unary function symbols.

Theorem 2 (Church)**.** *Let $\mathcal{L}$ be a non-trivial language. Then the collection* Taut($\mathcal{L}$) *of all tautologies in $\mathcal{L}$ is undecidable.*

Proof. If $\mathcal{L} \subseteq \{R\}$ with R binary then we can find a strongly undecidable $\mathcal{L}$-structure using the strongly undecidable graph $\mathcal{G}$ (How?).

If $\mathcal{L} \subseteq \{f\}$ with f binary then we can find a strongly undecidable $\mathcal{L}$-structure by using the previous example and using f as the interpretation of the characteristic function χ_R of R.

If $\mathcal{L} \subseteq \{d, r\}$ with d, r unary then we can find a strongly undecidable $\mathcal{L}$-structure as follows. Let $\mathcal{G} = (V, E)$ be a strongly undecidable graph and set $B = V \cup E$. For every $a \in V$ set $d(a) = r(a) = a$ and for every $(a, b) \in E$ let $d((a, b)) = a$ and $r((a, b)) = b$. This determines an $\{d, r\}$-structure $\mathcal{B} = (B, d^B, r^B)$ which can be extended to an $\mathcal{L}$ structure (by arbitrarily defining the interpretations of the remaining symbols). Notice that "V" is definable by the formula $d(x) = x$ and E is definable by the formula

$$\varphi(x, y) \equiv \exists z(\neg d(z) = z \wedge d(z) = x \wedge r(z) = y).$$

□

2. The word problem

2.1. **Intro.** A **semigroup** is a set S together with a binary associative operation $\cdot\colon S \times S \to S$. We will consider here monoids which moreover have an **identity element**, i.e., some $1 \in S$ with $1 \cdot s = s \cdot 1 = s$, for all $s \in S$ (these are usually called monoids). A semigroup S is a **group** if it is endowed additionally with a unary operation $s \mapsto s^{-1}$, the **inverse**, so that $s \cdot s^{-1} = s^{-1} \cdot s = 1$ for all $s \in S$. We will usually denote groups by Γ. In most cases one defines a semigroup by providing a presentation

$$S = \langle a_1, \ldots, a_n \mid R_1, \ldots R_k \rangle,$$

where R_i is an expression of the form $w_i = v_i$, with $w_i, v_i \in \{a_1, \ldots, a_n\}^*$. The semigroup defined by such presentation is the quotient $A^*/\sim$ of $A^* = \{a_1, \ldots, a_n\}^*$ via the the equivalence relation that is "spanned" by the relations $R_1, \ldots R_k$ (see Definition **??**). Similarly the group defined by the presentation

$$\Gamma = \langle a_1, \ldots, a_n \mid R_1, \ldots R_k \rangle,$$

is the quotient $(A \cup A^{-1})^*/\sim$ of $(A \cup A^{-1})^* = \{a_1, \ldots, a_n, a_1^{-1}, \ldots, a_n^{-1}\}^*$ via the the equivalence relation that is "spanned" by the relations $R_1, \ldots R_k$ and the relations $a_i^{-1} a_i = 1 = a_i a_i^{-1}$ for all $i \leq n$. Here are some examples (draw pictures):

$$F_2^{semi} = \langle a, b \rangle = \text{ all finite words in alphabet } \{a, b\} \text{ under composition.}$$

$$A_2^{semi} = \langle a, b \mid ab = ba \rangle = \text{ the abelian version of the above.}$$

Similarly we have the group versions of these two. For example $F_2 = \langle a, b \rangle$ is the collection of all words in a in alphabet $\{a, b, a^{-1}, b^{-1}\}$ where any instance of $a^{-1}a, aa^{-1}, b^{-1}b, bb^{-1}$ is collapsible to the empty sequence θ, and similarly for A_2.

The word problem. Given a presentation $\langle a_1, \ldots, a_n \mid R_1, \ldots R_k \rangle$ of a semigroup find an algorithm which in input $(w, v) \in A^* = \{a_1, \ldots, a_n\}^*$ decides whether $w \sim v$. Similarly for presentations of groups.

For example, in F_2^{semi} we have that $w \sim v$ iff $w = v$. For A_2^{semi} one can easily implement an algorithm bringing every word w in the form $(a^{k_w} b^{l_w})$ and then one sees that $w \sim v$ iff $k_w = k_v$ and $l_w = l_v$. Here is another example. Consider the group

$$\Gamma = \langle a, b \mid ab = ba, a^3 = 1 \rangle.$$

Again there is a simple algorithm which, given $w \in (A \cup A^{-1})^*$, it finds a word equivalent to w, of the form $(a^{k_w} b^{l_w})$ where $k_w \in \{0, 1, 2\}$ and $w \sim v$ iff $k_w = k_v$ and $l_w = l_v$.

$$aabab^{-1}ab \to aaabb^{-1}ab \to aaaab \to ab.$$

We have the following theorem.

Theorem 3 (Post, Markov (1947))**.** *There are semigroups for which the word problem is undecidable*

Of course this does not directly imply that the word problem for groups is undecidable in general. For example in the case of groups problem of deciding, given any w, v, whether $w \sim v$ reduces to the seemingly simpler problem of deciding whether $w' \sim 1$ for any given w'. Indeed the proof f the next theorem is significantly harder:

Theorem 4 (Novikov (1955), Boone (1959))**.** *There are groups for which the word problem is undecidable.*

Our first task is to prove Theorem 4 by reducing it to the more general problem of solving semi-Thue processes and showing that this latter one is undecidable.

2.2. **Semi-Thue processes and the word problem for semigroups.** Let $A = \{a_1, \ldots, a_n\}$ be a finite alphabet. A **semi-Thue production** is a pair of words $(w, \bar{w})$ which we denote by $w \to \bar{w}$. A**semi-Thue process** is a finite list Π of productions P. If P is the production $w \to \bar{w}$ and $u, v \in A^*$ then we write

$$u \Rightarrow_P v \text{ whenever } \exists x, y \in A^* (u = xwy \wedge v = x\bar{w}y)$$

Write $u \Rightarrow_\Pi v$ whenever $\exists p \in \Pi (u \Rightarrow_P v)$. Finally we write $u \Rightarrow_\Pi^* v$ if there is a sequence of productions leading from u to v.

Word problem for Π. Is there an algorithm which decides, given $u, v \in A^*$, whether $u \Rightarrow_\Pi^* v$? If so, we say that Π is solvable. Otherwise it is unsolvable.

Exercise.

(1) If $A = \{a\}$ then any production system Π on A is solvable.

(2) In any alphabet A, if Π consists of one production then Π is solvable.

Theorem 5. *There is an unsolvable semi-Thue process* Π *is some alphabet* A.

Proof. For every Turing machine M we define a semi-Thue process Π_M with the property that the halting problem for M reduces to the word problem for Π_M.

It suffice to consider Turing machines in alphabet $\{1\}$. Let M be such a Turing machine and use the symbol 0 to represent the empty squares of the tape. Let $Q_0, \ldots, Q_N$ be the collection of all states of M where Q_0 is the initial state and Q_N is the terminal state and recall that M is entirely determined by a finite set of **instructions**, which are 5-tuples of the form (Q_i, a, b, D, Q_j), with $a, b \in \{0, 1\}$ and $D \in \{\text{Left}, \text{Right}\}$, so that b, D, Q_j are uniquely determined by Q_i, a.

We define a semi-Thue process Π_M in the alphabet $A = \{0, 1, Q_0, \ldots, Q_N, Q, Q', \#\}$. We will use words of the form $\#uQ_iv\#$ with $u, v \in \{0, 1\}^*$ to represent the situation description of M which is in Q_i state and has the string uv in the (so far explored or written)tape with the read-write head in the first letter of v. Π_M contains precisely all productions from the following four categories:

(1) For every instruction of the form $(Q_i, a, b, \text{Right}, Q_j)$ include:

$$Q_ia0 \to bQ_j0 \qquad Q_ia1 \to bQ_j1 \qquad Q_ia\# \to bQ_j0\#$$

(2) For every instruction of the form $(Q_i, a, b, \text{Left}, Q_j)$ include:

$$0Q_ia \to Q_j0b \qquad 1Q_ia \to Q_j1b \qquad \#Q_ia \to \#Q_j0b$$

(3) Include $Q_N0 \to Q0$ and $Q_N1 \to Q1$

(4) Include $Q0 \to Q, \quad Q1 \to Q, \quad Q\# \to Q'\#, \quad 0Q' \to Q', \quad 1Q' \to Q'$

Notice now that all possible initial states of M correspond to the words:

$$\#Q_00\# \qquad \#Q_01\# \qquad \#Q_011\# \quad \cdots \quad \#Q_01^{(n)}\# \quad \cdots$$

where $1^{(n)}$ stands for n-many 1's. If M halts at the input $1^{(n)}$ then it is immediate that for some $u, v \in \{0, 1\}^*$ we have that

$$\#Q_01^{(n)}\# \;\Rightarrow_\Pi^* \; \#uQ_Nv\# \;\Rightarrow_{(3)}\; \#uQv\# \;\Rightarrow_\Pi^*\; \#uQ\# \;\Rightarrow_\Pi^*\; \#Q'\#.$$

Claim. *Conversely, if* $\#Q_01^{(n)}\# \;\Rightarrow_\Pi^*\; \#Q'\#$ *then* M *halts at the input* $1^{(n)}$.

Proof. The only way to reach $\#Q'\#$ uses productions of the type (3),(4). Hence, by backtracking we see that $\#Q_01^{(n)}\# \;\Rightarrow_\Pi^*\; \#uQ_Nv\#$, for some $u, v \in \{0, 1\}^*$. Notice that this later production can only use moves of type (1),(2) which correspond precisely to the computation of M and since Q_N is terminal M terminates for the desired input. □

Let now M be any Turing machine whose halting problem is not decidable (for example the universal Turing machine). The semi-Thue process Π_M for this M will not be solvable. □

2.3. **Thue processes.** To each production $w \to \bar{w}$ we associate its **inverse** $\bar{w} \to w$. A **Thue process** Π is a semi-Thue process which is closed under inverses. We can therefore write $w - \bar{w}$ without indicating direction and as a consequence $\Rightarrow_\Pi^*$ induces an equivalence relation $u \sim_\Pi v$ on A^*. We write $[u] = [u]_\Pi = \{v \in A^* \mid u \sim v\}$ for the equivalence of u and set $S = S_\Pi = A^*/\sim$ be the collection $\{[u] \mid u \in A^*\}$. Notice that concatenation in A^* pushes forward to a multiplication $\cdot$ on S since

$$u \sim_\Pi u' \quad \text{and} \quad v \sim_\Pi v' \quad \text{imply} \quad uv \sim_\Pi u'v'$$

So $(S, \cdot)$, with $[u] \cdot [v] = [uv]$ is a semigroup which has $[\theta]$ as an identity element. Conversely notice that every semigroup presentation corresponds to a Thue process Π and we set S_Π to be the semigroup defined by the presentation. This formalizes what we previously called "semigroup defined by the presentation".

Theorem 6. *There is a semigroup S with finite presentation whose word problem is undecidable.*

Proof. Recall that in the proof of Theorem 5 we associated to each M a semi-Thue process Π_M. Let Π_M^* be the Thue process resulting from additionally including all inverse productions of Π_M. With the notation of the proof of Theorem 5 we have:

Claim. $\#Q_0 1^{(n)}\# \;\Rightarrow_{\Pi_M}\; \#Q'\#$ *if and only if* $\#Q_0 1^{(n)}\# \;\sim_{\Pi_M^*}^*\; \#Q'\#$.

Proof. The $\Rightarrow$ direction is obvious. For the other assume that $\#Q_0 1^{(n)}\# \sim_{\Pi_M^*}^* \#Q'\#$ and let

$$w_0 \Rightarrow_{P_0^*} w_1 \Rightarrow_{P_1^*} \cdots \Rightarrow_{P_{n-1}^*} w_n$$

be a witness to this, i.e. $w_0 = \#Q_0 1^{(n)}\#$, $w_n = \#Q'\#$ and $P_i^* \in \Pi_M^*$ for all i. We will extract from this a production in Π_M from w_0 to w_n.

Let $i \leq n-1$ be the largest possible i with the property that $P_i^* \notin \Pi_M$. Notice that i has to be strictly less than $n-1$ because there is no production in Π_M which transforms $\#Q'\#$. Consider now the subsequence

$$w_i \Rightarrow_{P_i^*} w_{i+1} \Rightarrow_{P_{i+1}^*} w_{i+1}$$

and notice that by our assumptions we have that $P := (P_i^*)^{-1}$ and$R := P_{i+1}^*$ are in Π_M. So we have the following diagram in Π_M:

$$w_i \Leftarrow_P w_{i+1} \Rightarrow_R w_{i+1}$$

But notice now in each of the $w_0, \ldots, w_n$ there is a unique appearance of some "Q" symbol and by looking carefully at the rules of Π_M we see that there is a unique rule applying to w_{i+1}, i.e. $P = R$. We therefore have that $w_i = w_{i+2}$.

Inductively we can therefore remove all P_i^* which are not in Π_M. □

By the claim (and the argument in Theorem 5) the proof of the theorem follows. □

2.4. **Parenthesis: origin constrain domino tilling problem.** By a **domino type** we mean a square of unit size with each edge labeled by some letter. Let $\mathcal{D}$ be a finite set of domino types and let $d_0 \in \mathcal{D}$. A $(\mathcal{D}, d_0)$**-tilling** of $\mathbb{R}^2$ is a way of covering the entire $\mathbb{R}^2$ by squares of type $\mathcal{D}$ so that

(1) the vertexes of each domino are placed on the lattice $\mathbb{Z}^2$;
(2) a domino of type d_0 is placed in the origin $\{(0,0),(0,1),(1,0),(1,1)\}$
(3) adjacent edges have the same labels.

Is there an algorithm which, given finite $\mathcal{D}$ and d_0, decides whether there is a $(\mathcal{D}, d_0)$-tilling?

2.5. **The word problem for groups.** Recall that a **group presentation** consists of finite **alphabet** $A = \{a_1, \dots, a_n\}$ together with a finite collection $\mathcal{R}$ of **relations** $R_1, \dots R_k$ where R_i is an expression of the form, $w_i = \bar{w}_i$ with $w_i, \bar{w}_i \in A^*$. The group $\Gamma = \langle a_1, \dots, a_n \mid R_1, \dots R_k \rangle$ generated by the presentation is the semigroup associated to the following Thue process $\Pi^{\text{grp}}_{A,\mathcal{R}}$ on alphabet $A \cap A^{-1} = \{a_1, \dots, a_n\} \cup \{a_1^{-1}, \dots, a_n^{-1}\}$. We include the following two types of productions: for every $a_i \in A$ we add the productions $a_i a_i^{-1} - \theta$ and $a_i a_i^{-1} - \theta$; for every relation $R_i \equiv w_i = \bar{w}_i$ in the presentation we add the production $w_i - \bar{w}_i$. See subsection 2.3 for details. The main theorem of this subsections is the following:

Theorem 7 (Novikov, Boone)**.** *There is a group presentation whose word problem is undecidable, i.e., there is a presentation $A, \mathcal{R}$ whose associated Thue process $\Pi^{\text{grp}}_{A,\mathcal{R}}$ has undecidable word problem.*

For the proof of this theorem we will nee to: review some standard group theoretic constructions such as free products and HNN constructions; introduce *modular machines* as models of computation and show that one can reduce one them the halting problem; show that the halting problem for modular machine reduces to the word problem for presentations of groups.

We start by reviewing some group theoretic constructions. Let Γ_A, Γ_B we two groups with presentations $A = \{a_1, \dots, a_n\}, R_1, \dots, R_k$ and $B = \{b_1, \dots, b_m\}, S_1, \dots, S_l$ respectively. Assume that $A \cap B = \emptyset$. Their **free product** is simply the group

$$\Gamma_A * \Gamma_B := \langle a_1, \dots, a_n, b_1, \dots, b_m \mid R_1, \dots R_k, S_1, \dots, S_l \rangle.$$

Since $A \cap B = \emptyset$, it is not difficult to see that every word in $A \cup B$ is equivalent in $\Gamma_A * \Gamma_B$ to a word of the form $w_0 w_1 \cdots w_N$ where w_i is either a word in A or a word in B with A, B alternating from i to $i+1$ and for every $i > 0$ we have that $w_i \neq \theta$. In particular the free group in two generators $\langle a, b \rangle$ is the special case $\mathbb{Z} * \mathbb{Z}$ of this construction. Geometrically one can think of $\Gamma_A * \Gamma_B$ as tree (draw tree).

Recall that a group homomorphism is a map $\varphi \colon \Gamma \to \Delta$ between groups with $\varphi(gh) = \varphi(g)\varphi(h)$. It is an **isomorphism** if φ is bijective. An isomorphism from Γ to itself is called an **automorphism**. A special type of an automorphism $\Gamma \to \Gamma$ is given by conjugating by a fixed element $h \in \Gamma$, i.e. $g \mapsto h^{-1}gh$. These are precisely the **inner automorphisms** of Γ. Let now H, K be subgroups of Γ and let $\varphi \colon H \to K$ be an isomorphism of Γ. One often cares about the case when φ can be "implemented" by a global symmetry of the group, i.e., one often would like to be able to find some automorphism $\tilde{\varphi}$ of Γ which extends φ. Of course this is not always possible but one following construction shows that if we are allowed to extend Γ to a slightly larger group Γ_φ then we can find, in fact, an inner automorphism of Γ_φ which extends φ.

Let $\Gamma = \langle a_1, \dots, a_n \mid R_1, \dots, R_k \rangle$ and let $\varphi \colon H \to K$ be an isomorphism between subgroups of Γ. The **HNN extension of Γ with respect to φ** is the group

$$\Gamma_\varphi = \langle a_1, \dots, a_n, t \mid R_1, \dots, R_k, (t^{-1}ht = \varphi(h))_{h \in H} \rangle$$

In fact it suffice to include a relation $t^{-1}ht = \varphi(h)$ for every h in a fixed generating set of H. We call t the **stable letter** of the HNN extension. We will need two facts regarding HNN extensions which we will take for granted for now and later derive both of them from a common lemma.

Lemma 8. *The homomorphism $g \mapsto g$ from Γ to Γ_φ is injective, i.e., $\Gamma \subseteq \Gamma_\varphi$.*

Proof. Postponed for the end of the section. □

A subgroup L of Γ is **balanced with respect to** φ if $\varphi(L \cap H) = L \cap K$.

Lemma 9. *If L is a balanced subgroup of Γ then the subgroup L' of Γ_φ generated by L and t is the HNN extension $L_{\varphi\restriction L\cap H}$ of L with respect to $\varphi\restriction L \cap H$ and $L' \cap \Gamma = L$.*

Proof. Postponed for the end of the section. □

Sketch some topological perspective on presentations, free products and HNN extensions. Van Kampen theorem. Wedge sum. Adding mapping cylinders.

Next we introduce modular machines as a model of computation. This is an alternative to Turing machines that is slightly less handy for performing computations but it has the advantage of being easily reducible to the word problem. A **modular machine** $\mathcal{M}$ consists of a natural number $M > 1$ together with a finite set of **instructions**, i.e., quadruples of the form (a, b, c, D) where $a, b < M$, $c < M^2$, and $D \in \{L, R\}$. We assume that for every (a, b) there is at most one instruction (a, b, c, D). A **configuration** for a modular machine is just a pair $(\alpha, \beta) \in \mathbb{N}^2$. We write $(\alpha, \beta) \to_{\mathcal{M}} (\alpha', \beta')$ if and only if $\alpha = uM + a$ and $\beta = vM + b$ and there exists c so that either:

(1) $(a, b, c, R) \in \mathcal{M}$ and $\alpha' = uM^2 + c$ and $\beta' = v$; or
(2) $(a, b, c, L) \in \mathcal{M}$ and $\alpha' = u$ and $\beta' = vM^2 + c$;

We write $(\alpha, \beta) \to^*_{\mathcal{M}} (\alpha', \beta')$ if there is a finite sequence

$$(\alpha, \beta) = (\alpha_0, \beta_0) \to_{\mathcal{M}} (\alpha_1, \beta_1) \to_{\mathcal{M}} \cdots \to_{\mathcal{M}} (\alpha_n, \beta_n) = (\alpha', \beta')$$

A better way to think about modular machines is as follows: if (α, β) is a configuration then write $\alpha = \Sigma_{i=0}^{k} \alpha_i M^i$ and $\beta = \Sigma_{i=0}^{l} \beta_i M^i$ in M-ary expansion and think of α and β as being the two lists:

$$(\alpha_0, \alpha_1, \alpha_2, \ldots, \alpha_k) \quad \& \quad (\beta_0, \beta_1, \beta_2, \ldots, \beta_l)$$

If (a, b, c, R) applies to (α, β) then write $c = c_0 + c_1 M$ and notice that (α', β') corresponds to

$$(c_0, c_1, \alpha_1, \alpha_2, \ldots, \alpha_k) \quad \& \quad (\beta_1, \beta_2, \ldots, \beta_l).$$

Similarly, if (a, b, c, L) applies to (α, β) then (α', β') corresponds to

$$(\alpha_1, \alpha_2, \ldots, \alpha_k) \quad \& \quad (c_0, c_1, \beta_1, \beta_2, \ldots, \beta_l).$$

Theorem 10. *There exists a modular machine $\mathcal{M}$ so that the halting set*

$$H(\mathcal{M}) := \left\{(\alpha, \beta) \to^*_{\mathcal{M}} (0, 0)\right\}$$

is not recursive.

Proof. Let T be a Turing machine so that the set of all halting positions if not recursive. By adding some additional code to T we can make sure that whenever T halts the tape of T is empty.

Use the numbers $0, 1, \ldots, N-1$ to represent states of T and the numbers $N, N+1, \ldots, M-1$ to represent the alphabet of the tape of T. To each situation description

$$a_k\ a_{k-1}\ \ldots\ a_1\ Q\ a\ b_1\ b_2\ \ldots\ b_{l-1}\ b_l$$

where the head is on the cell containing a and T is in state Q associate the following two configurations of $\mathcal{M}$:

$$\text{Configuration 1.} \qquad (Q, a_1, a_2, \ldots, a_k) \quad \& \quad (a, b_1, b_2, \ldots, b_l).$$

$$\text{Configuration 2.} \qquad (a, a_1, a_2, \ldots, a_k) \quad \& \quad (Q, b_1, b_2, \ldots, b_l).$$

For every rule (Q, a, b, D, Q') of T add in $\mathcal{M}$ the instructions $(a, Q, Q' + bM, D)$ and $(Q, a, Q' + bM, D)$. Notice for example if (Q, a, b, R, Q') is a rule in T and the Turing is in the above situation then the next situation would be

$$a_k\ a_{k-1}\ \ldots\ a_1\ b\ Q'\ b_1\ b_2\ \ldots\ b_{l-1}\ b_l$$

which is corresponds to the configurations

$$\text{Configuration 1}'. \qquad (Q', b, a_1, a_2, \ldots, a_k) \quad \& \quad (b_1, b_2, \ldots, b_l).$$

$$\text{Configuration 2}'. \qquad (b_1, b, a_1, a_2, \ldots, a_k) \quad \& \quad (Q', b_2, \ldots, b_l).$$

We leave now to the reader to wrap up the proof. □

Next we describe the construction of a finite group presentation whose word problem is not decidable. The plan is to start with the group

$$G := \langle t, x, y \mid xy = yx \rangle = \mathbb{Z} * \mathbb{Z}^2$$

and extend this presentation by taking finitely many HNN extensions, in a way that the halting problem for $\mathcal{M}$ in the theorem above reduces to the word problem for the resulting group presentation. For every state $(\alpha, \beta) \in \mathbb{N}^2$ consider the word:

$$t(\alpha, \beta) := x^{-\alpha} y^{-\beta} t x^{\alpha} y^{\alpha}$$

and let $T = \langle t(\alpha, \beta) \mid \alpha, \beta \in \mathbb{N} \rangle$ be the subgroup of G spanned by $\{t(\alpha, \beta)\}_{\alpha,\beta \in \mathbb{N}}$.

Claim. *T is freely generated by $\{t(\alpha, \beta) : \alpha, \beta \in \mathbb{N}\}$.*

Proof. Recall that a word $v_0 v_1 \cdots v_n$ in some alphabet $(A \cup A^{-1})^*$ is call reduced there is no $i < n$ so that $v_i = v_{i+1}^{-1}$ or $v_i^{-1} = v_{i+1}$. We need to show that any non-empty reduced word $v_0 v_1 \cdots v_n$ in $\big(\{t(\alpha, \beta) : \alpha, \beta \in \mathbb{N}\} \cup \{(t(\alpha, \beta))^{-1} : \alpha, \beta \in \mathbb{N}\}\big)^*$ is not equal to 1 in G. This follows by an easy induction on n. □

Having "coded" the state (α, β) by the word $t(\alpha, \beta)$ we proceed now to "code" the instructions of the machine $\mathcal{M}$. For every instruction (a, b, c, R) in $\mathcal{M}$ we will introduce an isomorphism ϕ between certain subgroups G and then pass to the HNN extension G_ϕ with stable letter r. Similarly for (a, b, c, L) in $\mathcal{M}$ we will introduce an isomorphism ψ between certain subgroups G and then pass to the HNN extension G_ψ with stable letter l. The fact that we can realize all these HNN extensions simultaneously is just a consequence of Lemma 9.

Let (a, b, c, R) in $\mathcal{M}$. We want this operation to "transform" every word of the form $t(uM + a, vM + b)$ to the word $t(uM^2 + c, v)$. That is, we want

$$x^{-uM} y^{-vM} (x^{-a} y^{-b} t x^{a} y^{b}) x^{uM} y^{vM}$$

to get "transformed" to

$$x^{-uM^2} y^{-v} (x^{-c} t x^{c}) x^{uM^2} y^{v}.$$

This transformation can be implemented by the assignment:

$$x^M \mapsto x^{M^2}, \quad y^M \mapsto y, \quad t(a, b) \mapsto t(c, 0).$$

This assignment describes an isomorphism $\varphi \colon G_{a,b}^{M,M} \to G_{c,0}^{M^2,1}$ between the subgroups

$$G_{a,b}^{M,M} = \langle x^M, y^M, t(a, b) \rangle, \quad G_{c,0}^{M^2,1} = \langle x^{M^2}, y, t(c, 0) \rangle$$

of G which are both isomorphic to $\mathbb{Z} * \mathbb{Z}^2$. Similarly if (a, b, c, L) is in $\mathcal{M}$ then we consider the isomorphism $\psi \colon G_{a,b}^{M,M} \to G_{0,c}^{1,M^2}$ between the subgroups

$$G_{a,b}^{M,M} = \langle x^M, y^M, t(a, b) \rangle, \quad G_{0,c}^{1,M^2} = \langle x, y^{M^2}, t(0, c) \rangle$$

of G, given by the assignment

$$x^M \mapsto x, \quad y^M \mapsto y^{M^2}, \quad t(a, b) \mapsto t(0, c).$$

Given a modular machine $\mathcal{M}$ defined by M and the collection of instructions

$$\{(a_i, b_i, c_i, R)\colon i \in I\} \cup \{(a_j, b_j, c_j, L)\colon i \in J\}$$

we define the group $G_{\mathcal{M}}$ given by the following presentation. For every $i \in I$ we introduce a new letter r_i and we let G_i be the HNN extension of G with respect to the corresponding φ_i described above and with stable letter the letter r_i. Similarly, for every $j \in J$ we introduce a new letter l_j and we let G_j be the HNN extension of G with respect to the corresponding ψ_j with stable letter the letter l_j. By Lemma 8 G is a subgroup of G_i and G_j for all i, j and by Lemma 9 we can realize all these HNN simultaneously in the following presentation where i, j range over I, J respectively

$$G_{\mathcal{M}} := \langle x, y, t, r_i, l_j \mid xy = yx,\ \ r_i^{-1} G^{M,M}_{a_i,b_i} r_i = \varphi_i(G^{M,M}_{a_i,b_i}),\ \ l_j^{-1} G^{M,M}_{a_j,b_j} l_j = \psi_j(G^{M,M}_{a_j,b_j})\rangle.$$

Let now $H_{\mathcal{M}}$ be the subgroup of $G_{\mathcal{M}}$ that is generated by the elements

$$\{r_i\colon i \in I\} \cup \{l_j\colon j \in J\} \cup \{t(\alpha, \beta)\colon (\alpha, \beta) \to^*_{\mathcal{M}} (0, 0)\}$$

and let $F_{\mathcal{M}}$ be the subgroup of $G_{\mathcal{M}}$ that is generated by the elements

$$\{r_i\colon i \in I\} \cup \{l_j\colon j \in J\} \cup \{t\}$$

Lemma 11. *We have that $H_{\mathcal{M}} = F_{\mathcal{M}}$.*

Proof. Trivially we have that $t = x^0 y^0 t x^0 y^0 = t(0, 0)$ and $(0, 0) \to^*_{\mathcal{M}} (0, 0)$. Hence we have that $H_{\mathcal{M}} \supseteq F_{\mathcal{M}}$.

Conversely, we show by induction on the length of the computation of $\to^*_{\mathcal{M}}$ that if $(\alpha, \beta) \to^*_{\mathcal{M}} (0, 0)$ then $t(\alpha, \beta) \in F_{\mathcal{M}}$. For the induction step assume that $(\alpha, \beta) \to_{\mathcal{M}} (\alpha', \beta')$ and that $t(\alpha', \beta') \in F_{\mathcal{M}}$. Assume that this transition is performed by an instruction of the form (a, b, c, R), say the i-th instruction (a_i, b_i, c_i, R). The observation now is that the subgroup of $G_{\mathcal{M}}$ generated by $\{t(\alpha, \beta)\colon (\alpha, \beta) \to^*_{\mathcal{M}} (0, 0)\}$ is a balanced subgroup of G (with respect to φ_i) and therefore, within $H_{\mathcal{M}}$ we still have $r_i^{-1} t(\alpha, \beta) r_i = t(\alpha', \beta')$. Hence, $t(\alpha, \beta) = r_i t(\alpha', \beta') r_i^{-1}$, and it follows by the induction hypothesis that $t(\alpha, \beta) \in F_{\mathcal{M}}$. □

The last lemma reduces the problem of checking whether (α, β) halts in $\mathcal{M}$ to checking whether $t(\alpha, \beta)$ as an element of $G_{\mathcal{M}}$ is actually an element of $F_{\mathcal{M}}$. The next trick allows us to reduce it further to the word problem of a group presentation.

Theorem 12. *There is a finitely presented group $G^*_{\mathcal{M}}$ with unsolvable word problem.*

Proof. Let $\mathcal{M}$ be the modular machine from Theorem 10 and let $G_{\mathcal{M}}$ be the associated group and $F_{\mathcal{M}}$ its subgroup, both described in the previous paragraphs. Let

$$G^*_{\mathcal{M}} := \langle G_{\mathcal{M}}, s \mid \ s^{-1} F_{\mathcal{M}} s \rangle$$

be the HNN extension of $G_{\mathcal{M}}$ with respect to the isomorphism $\mathrm{id}\colon F_{\mathcal{M}} \to F_{\mathcal{M}}$. Since $F_{\mathcal{M}}$ is finitely generated the above presentation is finite. Moreover, by Lemma 9 and the claim after the definition of G we have that $s^{-1} g s = g$ if and only if $g \in F_{\mathcal{M}}$. Thus

$$(\alpha, \beta) \to^*_{\mathcal{M}} (0, 0) \quad \iff \quad t(\alpha, \beta) = s^{-1} t(\alpha, \beta) s \ \text{ in } \ G^*_{\mathcal{M}}$$

□

We are left to prove Lemma 8 and Lemma 9. Both will follow from the normal form theorem for HNN extensions.

Let $\varphi\colon H \to K$ be an isomorphism between subgroups of the group Γ. Let also

$$\Gamma_\varphi = \langle \Gamma, t \mid t^{-1}Ht =_\varphi K \rangle$$

be the HNN extension of H with respect to φ. Fix a section $S_H \subset \gamma$ for the right cosets of H in Γ and a section $S_K \subset \gamma$ for the right cosets of K in Γ, i.e. $\Gamma = \sqcup_{g\in S_H} Hg$ and similarly $\Gamma = \sqcup_{g\in S_K} Kg$. We assume the elements $g \in S_H$ and $g' \in S_K$ representing H and K are equal to 1. A **normal form (w.r.t. S_H, S_K)** is a sequence

$$g_0 t^{\epsilon_1} g_1 t^{\epsilon_2} g_2 \cdots t^{\epsilon_n} g_n,$$

with $\epsilon_i \in \{-1, 1\}$ and $g_i \in \Gamma$, so that for all i we have

(1) if $\epsilon_i = -1$ then $g_i \in S_H$;
(2) if $\epsilon_i = 1$ then $g_i \in S_K$;
(3) $t^{\epsilon_i} g_i t^{\epsilon_{i+1}}$ is never equal to $t^\epsilon 1 t^{-\epsilon}$.

A variant of the following theorem is often referred to as Britton's lemma.

Theorem 13. *For every $\Gamma, H, K, \varphi, S_H, S_K$ as above, every word $w \in \Gamma * \langle t \rangle$ is equivalent in Γ_φ to a unique normal form.*

Proof. First we show that every such word w is equivalent in Γ_φ to some normal form. Since $t^{-1}Ht =_\varphi K$ holds in Γ_φ we have the following quasi-commuting relations:

$$t^{-1}h = \varphi(h)\, t^{-1} \quad \text{and} \quad t\, k = \varphi^{-1}(k)\, t$$

for every $h \in H$ and for every $k \in K$. As a consequence, given any word

$$w = \gamma_0 t^{\epsilon_1} \gamma_1 t^{\epsilon_2} \gamma_2 \cdots t^{\epsilon_{n-1}} \gamma_{n-1} t^{\epsilon_n} \gamma_n,$$

we can use these relations going right to left and transform gradually w to a normal form. For example if $\epsilon_n = 1$ then the second relation applies: write γ_n as kg_n where $g_n \in S_K$ and use the quasi-commutation to transform w to

$$w' = \gamma_0 t^{\epsilon_1} \gamma_1 t^{\epsilon_2} \gamma_2 \cdots t^{\epsilon_{n-1}} \gamma_{n-1} \varphi^{-1}(h) t^{\epsilon_n} g_n.$$

After we reach to the end we cancel out all possible instances of (3) and repeat the process. Every time we cancel out all possible instances of (3) the length of w strictly decreases so the process eventually terminates.

To show uniqueness we employ an idea of Artin & van der Waerden. Let W be the collection of all normal forms and let $S(W)$ be group of all permutations of the set W. We define an action of Γ_φ on W, i.e., a homomorphism $\Phi\colon \Gamma_\varphi \to S(W)$. For that it suffices to explain where t maps and where each g in Γ maps and make sure that all relations coming from $t^{-1}Ht =_\varphi K$, $\gamma\gamma^{-1} = 1 = \gamma^{-1}\gamma$, push forward to relations which hold in $S(W)$. If $g \in \Gamma$ set

$$\Phi(g)(g_0 t^{\epsilon_1} g_1 t^{\epsilon_2} g_2 \cdots t^{\epsilon_n} g_n) = g g_0 t^{\epsilon_1} g_1 t^{\epsilon_2} g_2 \cdots t^{\epsilon_n} g_n$$

When it comes to t we distinguish two cases. If $g_0 \in K$ and $t^{\epsilon_1} = t^{-1}$ then set

$$\Phi(t)(g_0 t^{\epsilon_1} g_1 t^{\epsilon_2} g_2 \cdots t^{\epsilon_n} g_n) = \varphi^{-1}(g_0) g_1 t^{\epsilon_2} g_2 \cdots t^{\epsilon_n} g_n.$$

Otherwise set

$$\Phi(t)(g_0 t^{\epsilon_1} g_1 t^{\epsilon_2} g_2 \cdots t^{\epsilon_n} g_n) = \varphi^{-1}(k) t g_0' t^{\epsilon_1} g_1 t^{\epsilon_2} g_2 \cdots t^{\epsilon_n} g_n,$$

where $g_0 = k g_0'$ with $g_0 \in S_K$.

Exercise 14. *Show that Φ is a homomorphism, i.e., show that $\Phi(t)$ (for $\Phi(g)$ it is immediate) is a permutation (hint: define an inverse) and show that all the relations from the HNN presentation push forward to relations of $S(W)$.*

Notice now that if $g_0 t^{\epsilon_1} g_1 t^{\epsilon_2} g_2 \cdots t^{\epsilon_n} g_n$ is in normal form then

$$\Phi(g_0 t^{\epsilon_1} g_1 t^{\epsilon_2} g_2 \cdots t^{\epsilon_n} g_n)(1) = g_0 t^{\epsilon_1} g_1 t^{\epsilon_2} g_2 \cdots t^{\epsilon_n} g_n.$$

So the elements in Γ_ϕ corresponding to two distinct normal forms map via Φ to different elements of $S(W)$ and therefore they are distinct elements. □

3. Decidable theories

Contrary to the theory of $(\mathbb{N}, 0, S, +, *, <)$, the theory of partial orders, or the theory of graphs there are many first order theories which are decidable. To study decidability one has, for starters to restrict to recursive languages. This is not a real restriction since we will mostly deal with finite language which are always recursive. A language $\mathcal{L} = \{R_i^{(n)}, \ldots, f_j^{(m)}, \ldots, c_k, \ldots\}$ is **recursive** if the set

$$\{(n, i, m, j, k) \mid R_i^{(n)}, f_j^{(m)}, c_k \in \mathcal{L}\}$$

is recursive. Given a recursive language it is not difficult to see that the collections of all formulas and of all sentences are recursive.

The most useful technique in showing that some theory T is decidable (i.e. recursive) is quantifier elimination.

3.1. **Elimination of quantifiers.**

Definition 15. An $\mathcal{L}$-theory T **admits elimination of quantifiers** if for every formula $\varphi(x_1, \ldots, x_n)$, there exists a quantifier-free formula $\varphi^*(x_1, \ldots, x_n)$ so that

$$T \vdash \varphi(x_1, \ldots, x_n) \iff \varphi^*(x_1, \ldots, x_n)$$

It is important in the above definition that φ^* is in the same set of variables with φ. There is one problem with this steaming from the way we defined first order logic: if $\mathcal{L}$ has no constant symbols then it has no quantifier free sentence and therefore if, for example, $\exists x \forall y q(x, y)$ is an $\mathcal{L}$-sentence (say in prenex normal form) then we will not be able to get read of both quantifiers. To resolve this we could have defined first order logic to contain the 0-ary predicates $\top$ and $\bot$ which are always true and false respectively $\mathcal{L}$-sentences for each $\mathcal{L}$.

Since we will often work with $\mathcal{L}$ that contains constants we will not do this. Instead whenever $\mathcal{L}$ has no constants and φ is a sentence we will allow in the definition above

φ^* to be either $\exists x(x = x)$ or $\exists x\neg(x = x)$ which function like $\top$ and $\bot$ but have quantifiers.

Of course when $\mathcal{L}$ contains no constants if T has quantifier elimination then T is complete since every $\mathcal{L}$-sentence is provably equivalent to either $\top := \exists x(x = x)$ or $\bot := \exists x\neg(x = x)$. However, even if $\mathcal{L}$ has constants, quantifier elimination reduces the validity of any sentence to the validity of a quantifier-free sentence which is much easier to deal with. In fact, showing that a theory T is decidable often reduces to the following two steps:

(1) show that there is an algorithm for reducing φ to φ^*;
(2) show that there is an algorithm for deciding "$T \vdash q$" for all quantifier free formulas.

Examples. Let $T_{\mathbb{R}}$ be the theory of $(\mathbb{R}, 0, 1, +, *, <)$ and consider the sentence

$$\exists x(ax^2 + bx + c = 0).$$

Then one can eliminate the quantifier $\exists$ since this sentence is $T_{\mathbb{R}}$-equivalent to

$$(b^2 - 4ac > 0) \vee (b^2 - 4ac = 0 \wedge a \neq 0) \vee (a = b = c = 0)$$

Similarly the sentence $\exists_1\exists_2(ax_1 + bx_2 = 0 \wedge cx_1 + dx_2 = 0)$ is equivalent to

$$ad - bc \neq 0$$

The following criterion for quantifier elimination, while very basic, it is quite useful.

Lemma 16 (QE-criterion 1)**.** *T admits elimination of quantifiers if and only if T admits elimination of quantifiers for all formulas of the form*

$$\exists y\big(\alpha_1(y, \bar{x}) \wedge \cdots \wedge \alpha_n(y, \bar{x})\big),$$

where $\alpha_i(y, \bar{x})$ is either an atomic formula or a negation of an atomic formula

Proof. Assume that T eliminates quantifiers for this special type of formulas. Then we prove inductively that it eliminates quantifiers for all formulas:

(1) if φ is quantifier-free then set $\varphi^* = \varphi$;
(2) $(\neg\varphi) := \neg\varphi^*$;
(3) $(\varphi \wedge \psi)^* := \varphi^* \wedge \psi^*$;
(4) To define $(\exists y\varphi)^*$ we proceed as follows. First let φ^* be the quantifier-free formula equivalent to φ which exists by inductive hypothesis. We can always write such φ^* in disjunctive normal form, i.e.,

$$T \vdash \varphi^* \iff \big((\alpha_1^1 \wedge \cdots \wedge \alpha_{n(1)}^1) \vee \cdots \vee (\alpha_1^k \wedge \cdots \wedge \alpha_{n(k)}^k)\big)$$

and therefore we have that

$$T \vdash \exists y\varphi^* \iff \exists y\big((\alpha_1^1 \wedge \cdots \wedge \alpha_{n(1)}^1) \vee \cdots \vee (\alpha_1^k \wedge \cdots \wedge \alpha_{n(k)}^k)\big)$$

But $\exists$ distributes over $\vee$ and therefore by assumption

$$T \vdash \exists y\varphi^* \iff \xi^1 \vee \cdots \vee \xi^k,$$

where ξ^i is the quantifier free formula equivalent to $\alpha^i_1 \wedge \cdots \wedge \alpha^i_{n(i)}$. Set $(\exists y\varphi)^* := \vee_i \xi^i$

□

We can now directly apply this criterion to the following example. Let $\mathcal{L} = \{<\}$ and consider the $\mathcal{L}$-theory of **dense linear orderings without endpoints** $\mathrm{DLO}_{\pm\infty}$. This theory is axiomatized by the following axioms

(1) $\forall x \forall y \big(x < y \implies (\neg y < x \wedge \neg x = y)\big)$
(2) $\forall x \forall y \forall z \big((x < y \wedge y < z) \implies x < z\big)$
(3) $\forall x \forall y (x < y \vee x = y \vee y < x)$
(4) $\forall x \forall y \big(x < y \implies \exists z (x < z \wedge z < y)\big)$
(5) $\forall x \exists y (x < y)$, $\forall x \exists y (y < x)$

Here (1)-(3) declare that $<$ is a linear ordering, (4) says that it is dense, and (5) that it has no endpoints. The prototypical example of a structure which satisfies this theory is $(\mathbb{Q}, <)$. Another example is $(\mathbb{R}, <)$.

Theorem 17. *The theory* $\mathrm{DLO}_{\pm\infty}$ *admits elimination of quantifiers.*

Proof. We will use the criterion provided by lemma 16. So consider any $\mathcal{L}$-formula

$$\varphi(\bar{x}) \equiv \exists y \big(\alpha_1(y, \bar{x}) \wedge \cdots \wedge \alpha_n(y, \bar{x})\big),$$

where $\alpha_i(y, \bar{x})$ is either an atomic formula or a negation of an atomic formula. The possible forms of $\alpha_i(y, \bar{x})$'s are:

$$(a < b), \quad (a = b), \quad \neg(a < b), \quad \neg(a = b),$$

where a and b can be either equal to y or any entry from $\bar{x}$. Notice that we can assume without loss of generality that $\varphi(\bar{x})$ uses only the first two forms. To see this, notice that $\mathrm{DLO}_{\pm\infty}$ proves that:

$$\neg(a < b) \iff \big((a = b) \vee (b < a)\big) \quad \text{and} \quad \neg(a = b) \iff \big((a < b) \vee (b < a)\big),$$

and since $\exists$ distributes over $\vee$ the formula φ is $\mathrm{DLO}_{\pm\infty}$-equivalent to a formula:

$$\exists y(\alpha^1_1 \wedge \cdots \wedge \alpha^1_n) \vee \exists y(\alpha^2_1 \wedge \cdots \wedge \alpha^2_n) \vee \ldots \quad \ldots \vee \exists y(\alpha^k_1 \wedge \cdots \wedge \alpha^k_n).$$

Where each α^i_j is either of the form $(a < b)$ or $(a = b)$. For example if $\alpha_1 \equiv \neg(a = b)$,

$$\mathrm{DLO}_{\pm\infty} \vdash \varphi \iff \exists y \big((a < b) \wedge \alpha_2 \wedge \cdots \wedge \alpha_n\big) \vee \exists y \big((b < a) \wedge \alpha_2 \wedge \cdots \wedge \alpha_n\big)$$

So we assume that φ involves only $(a < b)$ and $(a = b)$ where $a, b \in \{y, \bar{x}\}$. Moreover if y does not appear in some α_i we could remove it from the scope of the $\exists y$. Finally if $y = y$ appears somewhere we can remove it and if $y < y$ appears somewhere, axiom (1) implies that $\mathrm{DLO}_{\pm\infty} \vdash (\varphi \iff \bot)$. So we can assume that $\varphi(\bar{x})$ is of the form:

$$\exists y \big(x_{i_1} < y \wedge \ldots \wedge x_{i_k} < y) \wedge (y < x_{j_1} \wedge \ldots \wedge y < x_{j_l}) \wedge (y = x_{r_1} \wedge \ldots \wedge y = x_{r_m})\big)$$

But if the last parenthesis is non-empty, say $y = x_{r_1}$ is there, then we the formula is equivalent with the formula attained by removing $\exists y$ and replacing all instances of y with x_{r_1}. We can therefore assume that $\varphi(\bar{x})$ is of the form:

$$\exists y\big(x_{i_1} < y \wedge \ldots \wedge x_{i_k} < y) \wedge (y < x_{j_1} \wedge \ldots \wedge y < x_{j_l})\big)$$

If either of the two parenthesis is empty then by Axiom (5) we have that $\mathrm{DLO}_{\pm\infty} \vdash (\varphi \iff \top)$. Otherwise, by Axiom (4) we have that

$$\mathrm{DLO}_{\pm\infty} \vdash \left(\varphi(\bar{x}) \iff \bigwedge_{n\leq k, m\leq l} x_{i_n} < x_{j_m}\right).$$

□

Corollary 18. *Since $\mathcal{L}$ in $\mathrm{DLO}_{\pm\infty}$ contains no symbols the above theorem shows that every sentence σ is either equivalent to $\top$ or to $\bot$. In other words $\mathrm{DLO}_{\pm\infty}$ is complete. It is moreover decidable. One can see that either by Corollary 62 of the notes in math117b or from the fact that the QE procedure described above is algorithmic.*

3.2. **Model theoretic criteria for QE.** Lemma 16 can be thought of as a "syntactic" criterion for quantifier elimination. Here we develop a very useful "semantic" criterion. That is, a criterion that is implies QE for some theory T as long as the the collection of all models of T behave nicely under "extending" and "gluing." We start with a lemma.

Lemma 19. *Let T be an $\mathcal{L}$-theory and let σ be an $\mathcal{L}$-sentence. The following are equivalent:*

(1) there exists a quantifier free σ^* so that $T \vdash (\sigma \iff \sigma^*)$;
(2) for every two $\mathcal{L}$-structures $\mathcal{A}, \mathcal{B}$ with $\mathcal{A}, \mathcal{B} \models T$, if $\mathcal{A}, \mathcal{B}$ satisfy the same quantifier-free sentences, then $\mathcal{A} \models \sigma \iff \mathcal{B} \models \sigma$.

Proof. (1) $\implies$ (2) is clear.

(2) $\implies$ (1). Before we prove it in complete generality let see what does this say about the case where $\mathcal{L}$ has no constants. In that case, the assumption "if $\mathcal{A}, \mathcal{B}$ satisfy the same quantifier-free sentences" is always satisfied simply because there are no quantifier free sentences. So (2) becomes

$$\text{"for every } \mathcal{A}, \mathcal{B} \models T \text{ we have } \mathcal{A} \models \sigma \iff \mathcal{B} \models \sigma. \text{ "}$$

But then, by Gödel's completeness theorem we know that $T \vdash \sigma$ and therefore $T \vdash (\sigma \iff \top)$.

Let now $\mathcal{L}$ be arbitrary and assume (2). Consider the collection S of all quantifier-free sentences with $T \cup \{\sigma\} \vdash S$. Notice that it is enough to show that $T \cup S \vdash \sigma$ because then (since proofs are finite) there would be $\sigma_1, \ldots, \sigma_k \in S$ so that

$$T \vdash (\sigma_1 \wedge \ldots \wedge \sigma_k) \implies \sigma$$

But since $\sigma^* := \sigma_1 \wedge \ldots \wedge \sigma_k$ is quantifier free $\sigma^* \in S$ (why?) we would have

$$T \vdash \sigma^* \iff \sigma$$

Claim. *We have that $T \cup S \vdash \sigma$.*

Proof of claim. If this fails then by Gödel's completeness theorem there is a model $\mathcal{A}$ of $T \cup S$ so that $\mathcal{A} \models \neg\sigma$. Let $S_{\mathcal{A}} \supseteq S$ be the collection of all quantifier free formulas satisfied by $\mathcal{A}$. By assumption of (2), and since quantifier-free formulas are closed under negation, every $\mathcal{B}$ with $\mathcal{B} \models T \cup S_{\mathcal{A}}$ should also satisfy $\neg\sigma$. That is $T \cup S_{\mathcal{A}} \vdash \neg\sigma$ so as above there is a single $\sigma_{\mathcal{A}} \in S_{\mathcal{A}}$ with $T \vdash \sigma_{\mathcal{A}} \implies \neg\sigma$. Hence, $T \vdash \sigma \implies \neg\sigma_{\mathcal{A}}$ and therefore $\neg\sigma_{\mathcal{A}} \in S$ which contradicts that $\mathcal{A} \models S$. □

□

Before we provide the model-theoretic criterion for quantifier-elimination, here is an example of a theory T which has models infinitely models which pairwise disagree on the quantifier free sentences, yet we will see later that T admits elimination of quantifiers. Let $\mathcal{L} = \{0, 1, +, *\}$ and consider the theory ACF of all **algebraically closed fields**. This theory is axiomatized by:

(1) all axioms for fields;
(2) for every n the axiom $\forall x_0 \cdots \forall x_n \exists y (x_n \neq 0 \implies x_n y^n + \cdots + x_1 y + x_0 = 0)$

Notice that ACF is satisfied by both the algebraic closure $\overline{\mathbb{Q}}^{acl}$ of $\mathbb{Q}$ as well as the algebraic closure $\overline{\mathbb{F}_p}^{acl}$ of $\mathbb{F}_p$ for each prime p. Notice that the following quantifier free formula is satisfied only by $\overline{\mathbb{F}_p}^{acl}$ iff $p = q$, and for no q in the case of $\overline{\mathbb{Q}}^{acl}$:

$$\overbrace{1 + \cdots + 1}^{q} = 0$$

Consider now the following two properties for the collection of all models of T.

Definition 20. Let T be an $\mathcal{L}$-theory. We say that T **has the isomorphism property** if for every $\mathcal{A}, \mathcal{B}$ with $\mathcal{A}, \mathcal{B} \models T$ and every isomorphism $f_0 : \mathcal{A}_0 \to \mathcal{B}_0$, between substructures $\mathcal{A}_0, \mathcal{B}_0$ of $\mathcal{A}, \mathcal{B}$ (possibly empty, do not model T in general) this isomorphism extends to an isomorphism $f_1 \colon \mathcal{A}_1 \to \mathcal{B}_1$ between substructures $\mathcal{A}_1 \subseteq \mathcal{A}, \mathcal{B}_1 \subseteq \mathcal{B}$ with $\mathcal{A}_1, \mathcal{B}_1 \models T$.

Consider the example the structures $\mathcal{A} = (\mathbb{Q}, <)$ and $\mathcal{B} = (\mathbb{R}, <)$ of DLO_∞, let $\mathcal{A}_0, \mathcal{B}_0$ be empty and find f_1...

Or consider the language $\mathcal{L} = \{<, c_1, c_2, \ldots\}$ and consider the theory DLO_∞ together with the axioms $c_i > c_{i+1}$ for all i. Consider the previous structures where the constants span out the substructures $\mathcal{A}_0 = (\{1, 1/2, 1/3, \ldots\}, <)$ and $\mathcal{B}_0 = (\{-1, -2, -3, \ldots\}, <)$ and find f_1.

Definition 21. Let T be an $\mathcal{L}$-theory. We say that T **has the absorption property** if for every $\mathcal{A} \subseteq \mathcal{B}$ with $\mathcal{A}, \mathcal{B} \models T$, any formula $\varphi(\bar{x})$ of the form $\exists y(\alpha_1 \wedge \cdots \wedge \alpha_n)$ where α_i is atomic or negation of atomic, and every tuple $\bar{a}$ from $\mathcal{A}$, we have that

$$\mathcal{A} \models \varphi(\bar{a}) \iff \mathcal{B} \models \varphi(\bar{a})$$

The combination of these two model theoretic properties provides a criterion for quantifier elimination.

Theorem 22. *Let T be an $\mathcal{L}$-theory. If T has the absorption property and the isomorphism property then T admits quantifier elimination.*

Proof. Let $\varphi(\bar{x})$ of the form $\exists y(\alpha_1 \wedge \cdots \wedge \alpha_n)$ where α_i is atomic or negation of atomic. We will find $\varphi^*(\bar{x})$ so that $T \vdash \varphi \iff \varphi^*$.

We introduce a constant c_i for every x_i in $\bar{x}$ and consider the language $\mathcal{L}' = \mathcal{L} \bigcup \{c_1, \ldots, c_n\}$. We view T as an $\mathcal{L}'$-theory. Consider now the $\mathcal{L}'$-sentence $\sigma = \varphi(\bar{c})$. Since we have not added any axioms for the c_i's, notice that if we find a quantifier-free $\mathcal{L}'$-sentence σ^* so that $T \vdash \varphi(\bar{c}) \iff \sigma^*$ then we have $T \vdash \varphi(\bar{x}) \iff \varphi^*(\bar{x})$ for the unique quantifier-free $\mathcal{L}$-formula $\varphi^*(\bar{x})$ with $\sigma^* = \varphi^*(\bar{c})$.

So we will now focus on finding a quantifier-free $\mathcal{L}'$-sentence σ^* so that $T \vdash \varphi(\bar{c}) \iff \sigma^*$

Claim. *The $\mathcal{L}'$-theory T has the absorption property and the isomorphism property.*

Proof. It easily follows from the definitions and we leave it as an exercise. □

To find the desired σ^*, by Lemma 19 it suffice to show that for every two $\mathcal{L}'$-structures $\mathcal{A}, \mathcal{B}$ with $T \models \mathcal{A}, \mathcal{B}$ and which satisfy the same quantifier free sentences, we have that $\mathcal{A} \models \varphi(\bar{c}) \iff \mathcal{B} \models \varphi(\bar{c})$. So let such $\mathcal{A}, \mathcal{B}$ and consider the smallest substructures $\mathcal{A}_0$,$\mathcal{B}_0$ of $\mathcal{A}, \mathcal{B}$ respectively. These are precisely the structures which are generated by the constants of $\mathcal{L}'$. Since $\mathcal{A}, \mathcal{B}$ satisfy the same quantifier-free formulas, the map $f_0 \colon \mathcal{A}_0 \to \mathcal{B}_0$ defined by $t^{\mathcal{A}} \mapsto t^{\mathcal{B}}$ for every closed term t is an isomorphism. By the isomorphism property it extends to an isomorphism $f_1 \colon \mathcal{A}_1 \to \mathcal{B}_1$ with $\mathcal{A}_1 \subseteq \mathcal{A}, \mathcal{B}_1 \subseteq \mathcal{B}$ which satisfy T . In particular $\mathcal{A}_1 \models \varphi(\bar{c}) \iff \mathcal{B}_1 \models \varphi(\bar{c})$ and by the absorption property (since $\bar{c}$ names elements of $\mathcal{A}, \mathcal{B}$) we have

$$\mathcal{A} \models \varphi(\bar{c}) \iff \mathcal{A}_1 \models \varphi(\bar{c}) \iff \mathcal{B}_1 \models \varphi(\bar{c}) \iff \mathcal{B} \models \varphi(\bar{c})$$

□

We can now sketch the proof that the theory of algebraically closed fields defined above has quantifier elimination. From that we will deduce some completeness and decidability results.

Theorem 23. *The theory* ACF *of algebraically closed fields admits elimination of quantifiers.*

Sketch of proof. By the last theorem it suffice to verify that ACF satisfies the isomorphism and the absorption property.

Isomorphism property. Let $\mathcal{A}, \mathcal{B} \models$ ACF and let f_0 be an isomorphism between any substructures $\mathcal{A}_0, \mathcal{B}_0$ of $\mathcal{A}, \mathcal{B}$. Notice that if it happened that the characteristic of $\mathcal{A}, \mathcal{B}$ is 0 then $\mathcal{A}_0, \mathcal{B}_0$ are not even rings.

Step 1. Extend $\mathcal{A}_0, \mathcal{B}_0$ to substructures $\mathcal{A}_0', \mathcal{B}_0'$ of $\mathcal{A}, \mathcal{B}$ which are rings and f_0 to an isomorphism f_0' between them.

This is done as follows: for every $a, a' \in \mathcal{A}_0$ the formula $x+a = a'$ defines a unique element in $\mathcal{A}$ which may or may not be in $\mathcal{A}_0$. Add all such elements in $\mathcal{A}_0'$ and check afterwards that $\mathcal{A}_0'$ is a ring. Similarly for $\mathcal{B}_0$. Finally extend f_0 to f_0' by sending the unique solution of $x + a = a'$ in $\mathcal{A}$ to the unique solution of $x + f_0(a) = f_0(a')$ in $\mathcal{B}$. Check that f_0 is an iso implies f_0' is an iso.

Step 2. Extend $\mathcal{A}_0', \mathcal{B}_0'$ to substructures $\mathcal{A}_0'', \mathcal{B}_0''$ of $\mathcal{A}, \mathcal{B}$ which are fields and f_0' to an isomorphism f_0'' between them.

Similar to the previous step: for every $a, a' \in \mathcal{A}_0'$ with $a \neq 0$ define $\mathcal{A}_0''$ by adding all solutions to the formula $x * a = a'$...

Step 3. Extend $\mathcal{A}_0'', \mathcal{B}_0''$ to substructures $\mathcal{A}_1, \mathcal{B}_1$ of $\mathcal{A}, \mathcal{B}$ which satisfy ACF and f_0'' to an isomorphism f_1 between them.

Similar to the previous step: see any algebra book (or notes from Ma5).

Absorption property. Let $\mathcal{A}, \mathcal{B} \models \mathrm{ACF}$ with $\mathcal{A} \subseteq \mathcal{B}$ and let $\varphi(\bar{x})$ be a formula of the form $\exists y(\alpha_1 \wedge \cdots \wedge \alpha_n)$, where α_i is atomic or negation of atomic. Fix also a $\bar{a}$ from $\mathcal{A}$. Every α_i has one of the following two forms

$$t(\bar{x}, y) = s(\bar{x}, y) \quad \text{or} \quad \neg t(\bar{x}, y) = s(\bar{x}, y).$$

So there exist polynomials in one variable $f_1, \ldots, f_k$ and $g_1, \ldots, g_l$ with coefficients from $\mathcal{A}$ so that $, \mathcal{B}$ satisfy $\varphi(\bar{a})$ iff the following system has a solution in $\mathcal{B}$:

$$f_1(y) = 0, \ldots, f_k(y) = 0, \quad g_1(y) \neq 0, \ldots, g_l(y) \neq 0.$$

If $k > 0$ then since the coefficients of f_1 are from $\mathcal{A}$ any solution of the system within $\mathcal{B}$ lies actually in $\mathcal{A}$. So in this case we indeed have $\mathcal{A} \models \varphi(\bar{a}) \iff \mathcal{B} \models \varphi(\bar{a})$.

If $k = 0$ then we have a finite system of inequalities. Since each g_i has finitely many possible solutions then the system always has a solution in $\mathcal{A}$ (or $\mathcal{B}$, respectively) if $\mathcal{A}$ (or $\mathcal{B}$) is infinite. But an algebraically closed field is always infinite since if it was finite we could enumerate it $a_1, a_2, \ldots, a_N$ and construct a polynomial without a solution:

$$(x - a_1)(x - a_2) \cdots (x - a_N) + 1 = 0$$

□

Corollaries.

(1) Definable subsets of A in any model $\mathcal{A}$ of ACF are defined equivalently by quantifier-free formulas. As a consequence they are either finite or cofinite,i.e., ACF is minimal.

(2) While ACF is not complete, the theory ACF_p where p is prime or 0 is complete. By ACF_p we mean the theory ACF to together with the axioms declaring the characteristic of the field. To see this notice that since every sentence σ is provably equivalent to a quantifier free σ^* then $\mathrm{ACF}_0 \vdash \sigma$ iff $(\mathbb{Q}, 0, 1, +, *) \models \sigma^*$ and $\mathrm{ACF}_p \vdash \sigma$ iff $(F_p, 0, 1, +, *) \models \sigma^*$.

(3) ACF_0 axiomatizes $(\mathbb{C}, 0, 1, +, *)$.

(4) ACF_0 as well as ACF_p are decidable.

3.3. **Presburger arithmetic.** We have seen that $\mathrm{Th}(\mathbb{N}, 0, S, +, *, <)$ is undecidable. Similarly it was a homework problem last quarter that $(\mathbb{N}, 0, S, *, <)$ defines addition and therefore the theory of the later is also undecidable. Here we will show that if we remove $*$ rather than $+$ the complexity of the theory drops significantly.

By **Presburger Arithmetic** T_{Presb} we mean the theory of $(\mathbb{N}, 0, S, +, <)$.

Theorem 24. *Presburger arithmetic T_{Presb} is a decidable theory.*

In order to prove this theorem we will establish first a quantifier elimination type of result for T_{Presb}. We say type of result because T_{Presb} does not have QE as such. For example notice that the following formula has no equivalent quantifier free one:

$$\varphi(x) \equiv \exists y(y + y = x)$$

Exercise. Prove this fact.

As a consequence we will need to introduce a collection of new predicates which will replace some formulas which do not admit QE within T_{Presb}. For every $m \geq 1$ consider let $x \equiv_m y$ be the binary relation $x \equiv y \mod m$ in $\mathbb{N}$ and notice that this relation is already definable $(\mathbb{N}, 0, S, +, <)$ by the $\exists_1$

$$\varphi_m(x, y) \equiv \exists z\big((x = y + \overbrace{z + \cdots + z}^{m-\text{times}}) \vee (y = x + \overbrace{z + \cdots + z}^{m-\text{times}})\big)$$

Let $T^+_{\mathrm{Presb}} = \mathrm{Th}(\mathbb{N}, 0, S, +, <, \equiv_2, \equiv_3, \ldots, \equiv_m, \ldots)$. The following theorem shows that the only formulas which do not admit QE in T_{Presb} are essentially these congruences.

Theorem 25. $T^+_{\mathrm{Presb}} = \mathrm{Th}(\mathbb{N}, 0, S, +, <, \equiv_2, \equiv_3, \ldots)$ *admits quantifier elimination.*

Proof. We will use Lemma 16, i.e., we will show that any formula $\varphi(\bar{x})$ of the form

$$(*) \qquad \exists y(\alpha_1(\bar{x}, y) \wedge \cdots \wedge \alpha_N(\bar{x}, y))$$

with α_i atomic/negation of atomic, is equivalent to a quantifier free formula. First we will show that we can restrict our attention to certain special such φ.

Step 0. An easy induction shows that every term in variables $\bar{x}, y$ is of the form

$$ny + t = ny + (n_1x_1 + \cdots + n_kx_k + n')$$

where $ny = S(\cdots(S(y))\cdots)$ n-many times, $n = S(\cdots(S(0))\cdots)$ n-many times, and for the purposes of this proof it suffice to condense the parenthesis above to the term $t = t(\bar{x})$.

Step 1. Each α_i can take one of the following forms:

$$(**) \qquad t_1 = t_2,\ t_1 < t_2,\ t_1 \equiv_m t_2, \quad \text{or} \quad t_1 \neq t_2,\ t_1 \not< t_2,\ t_1 \not\equiv_m t_2.$$

The first observation is that we can re-express everything as a disjunction of expression like in $(*)$ so that each α_i there uses only the first three forms from $(**)$. To see this just use the fact that $\vee$ distributes over $\exists$ after you replace $t_1 \neq t_2$ with $(t_1 < t_2 \vee t_2 < t_1)$, $t_1 \not< t_2$ with $(t_2 < t_1 \vee t_1 = t_2)$, and $t_1 \not\equiv_m t_2$ with

$(\vee_{i=1}^{m-1}(t_1 \equiv_m t_2 + i))$. We can assume that y appears in each α_i (otherwise we can take it out of $(\exists y)$) so each α_i is in one of the forms:

$$ny + t = ly + s, \quad ny + t < ly + s, \quad ny + t \equiv_m ly + s.$$

By cancelling out y's we have that each α_i is in one of the forms:

$$ky + t = s, \quad ky + t < s, \quad t < s + ky, \quad ky + t \equiv_m s.$$

Step 2. Uniformize all the y-coefficients to py by multiplying each ky with p/k where p is the lower common multiple of all k coefficients. Here each $ky + t \equiv_m s$ will change to $py + t * (p/k) \equiv_{(p/k)*m} s * (p/k)$. So each α_i is in one of the forms:

$$py = \text{“}s - t\text{”}, \quad py < \text{“}s - t\text{”}, \quad \text{“}t - s\text{”} < py, \quad py \equiv_m \text{“}s - t\text{”}.$$

where p is common across all α_i and “$t - s$” $< py$ stands for $t < py + s$ (since we have no “$-$” we cannot literally write $t - s$ but it is easier to read this way). Finally we can replace py with a new variable z and add a new α_i at the end of the form $z \equiv_p 0$. So in other words we have reduced $(*)$ to an expression that involves only α_i of the form $z = s - t$, $z < s - t$, $z > s - t$, $z \equiv_m s - t$ (for readability we omit “ ”).

Step 3. If there is any α_i of the form $z = s - t$ then we are done: remove the quantifier $\exists z$ and replace all appearances of z with $s - t$. So we can assume that $(*)$ is of the form

$$\begin{aligned}\exists z\Big(\big((r_1 - s_1 < z) \wedge \cdots \wedge (r_l - s_l < z)\big) \wedge \\ \wedge \big((z < t_1 - u_1) \wedge \cdots \wedge (z < t_k - u_k)\big) \wedge \\ \wedge \big((z \equiv_{m_1} v_1 - w_1) \wedge \cdots \wedge (z \equiv_{m_n} v_n - w_n)\big)\Big)\end{aligned}$$

Case 1. If the third parenthesis is empty the above expression is equivalent to

$$\bigwedge_{i=1}^{l}\bigwedge_{j=1}^{k}\big((r_i - s_i + 1 < t_j - u_j)\big) \wedge \big(\bigwedge_{j=1}^{k}(0 < t_j - u_j)\big).$$

Case 2. Otherwise the solution to the system still has to be between $\max_i(r_i - s_i)$ and $\min_j(t_j - u_j)$ but moreover it has to satisfy the restrictions imposed by the third parenthesis. But to find any solution to the restrictions of the third parenthesis it suffice to scan any interval of length $M = \text{lcm}\{m_1, \ldots, m_n\}$ between $\max_i(r_i - s_i)$ and $\min_j(t_j - u_j)$. That is, we can replace $\exists$ with a $\bigvee$ bounded by M. The only catch is that we cannot express directly $\min, \max$ but instead we write the equivalent:

$$\bigvee_{e=0}^{l}\bigvee_{q\le M}\Big(\bigwedge_{i\le l}(r_i - s_i < r_e - s_e + q) \wedge \bigwedge_{j\le k}(r_e - s_e + q < t_j - u_j) \wedge \bigwedge_{p\le n}(r_e - s_e + q \equiv_{m_p} u_p - w_p)\Big),$$

where by r_0 and s_0 we mean 0 (this is used in case the first parenthesis in the $\exists z$ expression is empty.) □

We can now finish the proof of Theorem 24.

Proof of Theorem 24. Let σ be any sentence in the language $(0, S, +, <)$. The proof of 25 provides an algorithm for reducing σ to a quantifier free sentence σ^* of $(0, S, +, <, \equiv_2, \equiv_3, \ldots)$ so that $T_{\text{Presb}} \vdash \sigma \iff \sigma^*$ (here we replace $\equiv_m$ with $\varphi_m(x, y)$. The proof now follows from the following exercise:

Exercise. Describe an algorithm that decides the truth-value of every quantifier-free sentence σ^* of $(0, S, +, <, \equiv_2, \equiv_3, \ldots)$ when this sentence is evaluated in

$$(\mathbb{N}, 0, S, +, <, \equiv_2, \equiv_3, \ldots)$$

□

Corollary 26. *$A \subseteq \mathbb{N}$ is definable in $(\mathbb{N}, 0, S, +, <)$ if and only if A is* **eventually periodic***, i.e., if there exists M, p so that $\forall n > M(n \in A \iff n + p \in A)$.*

Proof. **HW** □

Corollary 27. *Multiplication is not definable in $(\mathbb{N}, 0, S, +, <)$.*

Proof. We know that $(\mathbb{N}, 0, S, +, *, <)$ defines all recursive (in fact all arithmetical sets) and clearly there are recursive sets which are not eventually periodic. □

Working backwards the proof of Theorem 25,24 we can isolate all the axioms necessary to decide all sentences true in $(\mathbb{N}, 0, S, +, *, <)$. In particular, $\mathrm{T}_{\text{Presb}}$ is completely axiomatizable by:

(1) $\neg 0 = 1$;
(2) $\forall x \forall y \forall z (x+y = y+x) \wedge (x+(y+z)) = ((x+y)+z) \wedge (x+z = y+z \implies x = y)$;
(3) $\forall x S(x) = x + S(0)$;
(4) $<$ is linear;
(5) $\forall x \forall y (x < y \iff \exists z (z \neq 0 \wedge y = x + z))$
(6) for all n the axiom: $\forall x \exists y \exists z \big(x = ny + z \wedge (z = 0 \vee z = 1 \vee \cdots \vee z = n-1)\big)$.

A similar approach shows that the theory of ordered Abelian groups is also decidable.

4. Computational complexity of decision problems

In Math117b we saw that there are various degrees of undecidability. Ranked with respect to computable reductions they form the collection of all Turing degrees $\{[\alpha]_T \colon \alpha \in \mathbb{N}^{\mathbb{N}}\}$. In this section we will develop a hierarchy of "tractability" within the collection of all decidable problems, i.e., of problems whose Turing degree is the smallest possible $\underset{\sim}{0}$. *Tractable problems* will be the ones which can be solved with an algorithm which terminates in polynomial time. The righe notion of reduction in this context is the *polynomial-time reduction*. Among other things we will see that the decision problem for sentences of Presburger arithmetic is *exponentially time hard*, i.e., highly untractable.

Let A be a finite alphabet. A **language** L is just any collection of words in A, i.e., any $L \subset A^*$. We denote by $|w|$ the length of the word w. Given a language L and any function $t \colon \mathbb{N} \to \mathbb{N}$ we say that L has **time complexity** t if there exists a

Turing machine M on some alphabet $B \supseteq A$ so that M has two distinguished states Q_Y, Q_N so that:

$$w \in L \implies \text{on input } w, M \text{ stops in at most } t(|w|) \text{ steps on state } Q_Y;$$

$$w \notin L \implies \text{on input } w, M \text{ stops in at most } t(|w|) \text{ steps on state } Q_N.$$

We say that M **decides** L **in time** t. Let $\mathcal{P}$ be the class of all **tractable languages**, or **polynomial time languages**, i.e. all L which are decided in time p where p is any polynomial. If $L \notin \mathcal{P}$ then we say that L is **intractable**.

Remark. In the proof of equivalence of the various models of computation (Math117a) all the reductions introduced at most polynomial time slowdown. As a consequence the above notion of tractability is robust.

Let now A, B be two alphabets and let $f\colon A^* \to B^*$. We say that f is a **polynomial-time function** if there exists a Turing machine M on some alphabet $\Sigma \supseteq A \cup B$ which always halts on input $w \in A^*$ and the output when it halts is $f(w)$. Let $L \subseteq A^*$ and $R \subseteq B^*$. We say that L is **polynomial-time reducible to** R and we write $L \leq_{\mathcal{P}} R$ if there exists a polynomial time function f with

$$w \in L \iff f(w) \in R.$$

If $L \leq_{\mathcal{P}} R$ then notice $R \in \mathcal{P}$ implies $L \in \mathcal{P}$, and if L is **intractable** then so is R.

Definition 28. Let $\mathcal{C}$ be a collection of languages and let $R \subseteq B^*$. Then we say that R is $\mathcal{C}$**-hard** if for every $L \in \mathcal{C}$ we have that $L \leq_{\mathcal{P}} R$. If additionally $R \in \mathcal{C}$ then we say that R is $\mathcal{C}$**-complete**.

Here we will mostly focus on three classes: the class $\mathcal{P}$ of polynomial time languages, the class $\mathcal{E} = \bigcup_{c>0} \mathrm{TIME}(2^{cn})$ of **exponential time languages**, and the class $\mathcal{NP}$ of *not-deterministic polynomial time languages* which we are about to define.

Definition 29. Let $L \subseteq A^*$. We say that L is a **non-deterministic polynomial time** language, and we write $A \in \mathcal{NP}$ if there exists a Turing machine M on some alphabet $B \supseteq A \bigcup \{\#\}$ and a polynomial $p = p(n)$ so that for every $w \in A^*$ we have that

$$w \in L \iff \exists v \in B^*(|v| < p(|w|)) \wedge M \text{ halts in } \leq p(|w|) \text{ steps on input } w\#v$$

We call the v above a **verifier** or **certificate** for w.

So $\mathcal{NP}$ languages are the ones for which there exists an algorithm which decides them in polynomial time as long as it is fed the right "mini oracle." Of course there are $|B|^{p(n)}$-many such "mini oracles" making $\mathcal{NP}$ languages potentially of exponential time. It is clear that any problem in $\mathcal{P}$ is also in $\mathcal{NP}$ since one can take the verifier to be θ in the above definition. One of the biggest open problems currently is:

$$\text{Prove or disprove that } \mathcal{P} \neq \mathcal{NP}.$$

Here are some examples of decision problems which are in $\mathcal{NP}$:

Example. CLIQUE
Instance: an undirected graph G (in the form of an adjacency matrix fed row by row) and a natural number N (fed in binary).
Question: does G have a clique of size N?

Example. PRIMES
Instance: a natural number $N > 1$ (fed in binary).
Question: is N prime?

Question: does G have a clique of size N?

Example. SAT
Instance: a finite set $V = \{v_1, v_2, \ldots, v_k\}$ of propositional variables (fed in the form vn where n binary) and a finite set $C = \{c_1, \ldots, c_m\}$ of clauses $c_i = \vee_{j=1}^{n_i} (v_j)^{\varepsilon_j}$ where the literals $(v_j)^{\varepsilon_i}$ are either variables or negations of variables.
Question: is there an assignment of truthvalues $V \to \{\top, \bot\}$ which makes

$$\bigwedge_{i=1}^{m} \bigvee_{j=1}^{n_i} (v_j)^{\varepsilon_j} \text{ true ?}$$

Definition 30. A language $L \subseteq A^*$ is in the class co–$\mathcal{NP}$ iff $A^* \setminus L \in \mathcal{NP}$.

Theorem 31 (Cook-Levin)**.** SAT *is $\mathcal{NP}$-complete.*

Proof. It is clearly in $\mathcal{NP}$. We leave this to the reader. So we are left to show that it is $\mathcal{NP}$-hard.

Let $L \subseteq A^*$ be in $\mathcal{NP}$ and let $D = \{v, 0, 1, |\}$ be the alphabet of SAT where $|$ is used to separate between clauses and, say $||$ will separate between the variables and clauses (there are many ways to do it and we will not be so precise). We will define a polynomial time $f\colon A^* \to D^*$ which reduces L to SAT. For that we will use B, M, p from the definition of $\mathcal{NP}$ to define the right variables and clauses.

So let $B \subseteq A \cup \{Q_Y, \#\}$, then Turing machine M and the polynomial $p = p(n)$ associated to the "verifier algorithm" for L. Let $Q_0, \ldots, Q_m$ be the states of M with Q_0 initial and Q_m terminal. We can also assume that all instructions which start with Q_m are of the form $(Q_m, b, b, \_, Q_m)$.

Now $w \in L$ with $|w| = n$ if and only if M halts after $p(n)$-many steps on input $w\#v$ with Q_Y on the tape. So the head of the tape will not visit any cell past the interval $[-p(n), p(n)]$. Let $A = \{\sigma_1, \ldots, \sigma_K\}$, $A = \{\sigma_1, \ldots, \sigma_K, \sigma_{K+1}, \ldots, \sigma_L\}$, and $\sigma_0 = \#$.

We associate to M the following variables which depend on n:

(1) $S_{i,k}$ where $0 \le i \le p(n), 0 \le i \le m$;
(2) $H_{i,j}$ where $0 \le i \le p(n), -p(n) \le j \le p(n)$;

(3) $T_{i,j,l}$ where $0 \leq i \leq p(n), 0 \leq i \leq m, 0 \leq l \leq L$.

Here $S_{i,k}$ will be true if Q_k is the state during the i-th step of the computation. $H_{i,j}$ will be true if the head is positioned on the j-th cell during the i-th step of the computation. $T_{i,j,l}$ will be true if at the i-th step of the computation, the j-th cell reads σ_l.

To each input $w = \sigma_{k_0} \cdots \sigma_{k_{n-1}} \in A^*$, $k_r \in [1, K]$ we associate the following clauses:

(1)
- $S_{i,0} \vee \cdots \vee S_{i,m}$ for every i (always in some state);
- $\neg R_{i,k} \vee \neg R_{i,k'}$ for every i and all $k \neq k'$ (cannot occupy both states)
- $H_{i,-p(n)} \vee \cdots \vee H_{i,p(n)}$ for every i (always in some cell);
- $\neg H_{i,j} \vee \neg H_{i,j'}$ for every i and all $j \neq j'$ (cell is unique)
- $T_{i,j,0} \vee \cdots \vee T_{i,j,l}$ for every i, j (scan at least one symbol);
- $\neg T_{i,j,l} \vee \neg T_{i,j,l'}$ for every i, j and all $l \neq l'$ (symbols scanned is unique);

(2)
- $S_{0,0}, H_{0,0}, T_{0,j,k_j}$ for $0 \leq j \leq n-1$, $T_{0,n,0}$ and $T_{0,j,0}$ if $j < 0$ (initial);

(3)
- $S_{p(n),m}$ (positive halting position).

(4) for every instruction $(Q_k, \sigma_l, \sigma_{l'}, s, Q_{k'})$ in M the right transitions, e.g. if $s = R$ then add for every i:
- $\neg S_{i,k} \vee \neg H_{i,j} \vee \neg T_{i,j,l} \vee S_{i+1,k'}$;
- $\neg S_{i,k} \vee \neg H_{i,j} \vee \neg T_{i,j,l} \vee H_{i+1,j+1}$;
- $\neg S_{i,k} \vee \neg H_{i,j} \vee \neg T_{i,j,l} \vee T_{i+1,j+1,l'}$;

So $f(w)$ constitutes the above collection variables (for $n = |w|$) and the above collection of clauses. Given B, M, p, all these data can be clearly extracted from w in polynomial time. We are left to show that

$$w \in L \iff f(w) \in \text{SAT}.$$

If $w \in L$ then M halts in $\leq p(|w|)$-time for input $w\#v$ and some some v with $|v| \leq p(|w|)$. Now feed the computation of M in the above circuit by making the associated variables true and notice that a terminating computation of M guarantees the satisfiability of the circuit (if $|v| + |w| + 1 > p(|w|)$ then we feed only the $(|v| - |w| - 1)$ -part of v in the $T_{0,j,l}$'s. This will not affect the computation since the tape cannot reach to the $p(|w|) + 1$ cell before time $p(|w|)$).

Conversely if the circuit is satisfiable then read from the $T_{0,j,l}$'s the right v and prove by an easy induction that M halts in $p(|w|)$-time (use (1)-(4) to read from the circuit each step of the computation). □

4.1. **More examples of $\mathcal{NP}$-complete problems.**

Rather than proving from scratch each time, as in the case of SAT, that a problem is $\mathcal{NP}$-complete, we can instead polynomially reduce on it some problem which we already know is $\mathcal{NP}$-complete. We provide here several examples where this technique applies.

Example. 3-SAT
Instance: as in SAT one is given a finite set V of propositional variables and a finite set C of clauses which are of the form $x^\varepsilon \vee y^\varepsilon \vee z^\varepsilon$.
Question: is there an assignment of truthvalues $\wedge_{i=1}^m (x_{n_i}^\varepsilon \vee y_{n_i}^\varepsilon \vee z_{n_i}^\varepsilon)$ true.

Theorem 32. *3-SAT is $\mathcal{NP}$-complete.*

Proof. It is clearly in $\mathcal{NP}$. The mini oracles v here are simply assignment of truthvalues for the variables. If one is given a v which witnesses that the circuit w is satisfiable then checking that this v indeed witnesses the satisfiability of w is of polynomial complexity (in fact linear complexity). So we are left with showing that it is $\mathcal{NP}$-hard. Since is SAT is $\mathcal{NP}$-hard it suffice to show:

$$\text{SAT} \leq_{\mathcal{P}} \text{3-SAT}.$$

For the reduction let w be a word in the alphabet of SAT if w does not even represent a circuit (variable set/clause set) let $f(w)$ (this checking takes polynomial time). Otherwise if w is a circuit we translate each clause of w to an equivalent collection of clauses of size 3, defining this way $f(w)$ as follows:
change each clause $(x \vee y)$ to the set $(x \vee y \vee z)$, $(x \vee y \vee \bar{z})$;
change each clause (x) to the set $(x \vee y \vee z)$, $(x \vee y \vee \bar{z})$, $(x \vee \bar{y} \vee z)$, $(x \vee \bar{y} \vee \bar{z})$;
change each clause $(x_1 \vee \cdots \vee x_n)$ to the set

$$(x_1 \vee x_2 \vee y_1),\ \ (x_3 \vee \bar{y}_1 \vee y_2), (x_4 \vee \bar{y}_2 \vee y_3), \ldots, (x_{n-2} \vee \bar{y}_{n-4} \vee y_{n-3}),\ \ (x_{n-1} \vee x_n \vee \bar{y}_{n-3})$$

We leave to the reader to check that these are equivalent circuits from the point of view of satisfiability and to confirm that the translation is done in polynomial time. □

Example. SUBGRAPH ISO
Instance: Two finite graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$.
Question: Does G_1 contain a subgraph isomorphic to G_2, i.e, is there a subset W_1 of V_1 and a subset F_1 of E_1 such that the graph (W_1, F_1) is isomorphic to G_2?

Theorem 33. SUBGRAPH ISO *is $\mathcal{NP}$-complete.*

Proof. As before SUBGRAPH ISO is in $\mathcal{NP}$. We show that it is $\mathcal{NP}$-hard. By the HW CLIQUE is $\mathcal{NP}$-hard so it suffice to show

$$\text{CLIQUE} \leq_{\mathcal{P}} \text{SUBGRAPH ISO}.$$

This is clear since an instance G, N of CLIQUE directly translates to the instance G_1, G_2 of SUBGRAPH ISO where $G_1 = G$ and G_2 is the complete graph K_N in N vertexes. □

Example. GRAPH COLOURABILITY
Instance: A graph $G = (V, E)$ and $k \leq |V|$.
Question: Is G k-colourable, i.e., can one assign to each vertex of G a number from $\{1, \ldots, k\}$ so that E-adjacent vertexes are assigned different colours?

Theorem 34. GRAPH COLOURABILITY *is $\mathcal{NP}$-complete.*

Proof. As before GRAPH COLOURABILITY is in $\mathcal{NP}$. We show now that it is $\mathcal{NP}$-hard by polynomially reducing on it 3-SAT.

Let $w = (X, C)$ be an instance of 3-SAT where $X = \{x_1, \ldots, x_n\}$ is a set of variables and $C = \{F_1, \ldots, F_m\}$ is a collection of 3-clauses. We map w to an instance (G, k) of GRAPH COLOURABILITY as follows. Set $k = n + 1$ and consider the graph G whose vertexes are (here the z_i constitute a new set of points):

$$\{z_1, \ldots, z_n\} \cup \{x_1, \ldots, x_n\} \cup \{\bar{x}_1, \ldots, \bar{x}_n\} \cup \{F_1, \ldots, F_m\}$$

The edge relations are all the pairs of the form: (z_i, z_j) for $i \neq j$; (z_i, x_j) for $i \neq j$; $(z_i, \bar{x}_j)$ for $i \neq j$; $(x_i, \bar{x}_i)$;$(x_i, F_j), (F_j, x_i)$ whenever $x_i \notin F_j$; $(\bar{x}_i, F_j), (F_j, \bar{x}_i)$ whenever $\bar{x}_i \notin F_j$. The assignment $w \mapsto f(w)$ sending (X, C) to the (G, k) above is clearly computed in polynomial time. We are left with showing that it is a reduction.

If (X, C) is satisfiable under the truth assignment $t \colon X \to \{\top, \bot\}$ then G is k-colourable: use the colours $\{1, \ldots, n\}$ to color the complete graph $\{z_1, \ldots, z_n\}$ and call the remaining color s. Look at $x_i, \bar{x}_i$. There are connected between them so they have to take different colours. Since they are connected with z_j for $i \neq j$ they only colours they can get is either i or s. If $t(x_i) = \top$ colour x_i with i and $\bar{x}_i$ with s. Otherwise do the opposite. Each F_l now is of the form $x_i^{\varepsilon_i} \vee x_j^{\varepsilon_j} \vee x_p^{\varepsilon_p}$ with one of the literals taking the truth value $\top$ under the assignment t. For the first such literal, say i, color F_l by i. We leave it to the reader to check that this works.

Conversely, if (G, k) is k-colourable then by permuting the colors we can assume that z_i is coloured by i. Call again s the remaining colour and consider the assignment $t \colon X \to \{\top, \bot\}$ where $t(x_i) = \top$ if and only if the color of x_i is i. We leave it to the reader to check that this works. □

Problem. TRAVELING SALESMAN
Instance: A collection $C = \{c_i, \ldots, c_m\}$ of cities, a distance function $d(c_i, c_j) \in \{1, 2, \ldots\}$ for $i \neq j$ and bound $B \in \{1, 2, \ldots\}$.
Question: Is there a tour of all the cities of total distance $\leq B$, i.e., an ordering $c_{\sigma(1)}, \ldots, c_{\sigma(m)}$ so that the sum of the distance between consecutive cities (including $d(c_{\sigma(m)}, c_{\sigma(1)})$) at the end) is $\leq B$?

It is clear that TRAVELING SALESMAN is in $\mathcal{NP}$. We will show that it is $\mathcal{NP}$-complete with a series of reduction.

Problem. HAMILTONIAN CIRCUIT
Instance: A graph $G = (V, E)$.
Question: Is there an (injective) ordering $v_1, \ldots, v_{n-1}$ of V so that $v_1, \ldots, v_{n-1}, v_n$ where $v_n = v_1$ is a path?

Theorem 35. HAMILTONIAN CIRCUIT *is polynomial time reducible to* TRAVELING SALESMAN.

Proof. Given $G = (V, E)$ define (C, d, B) by $C = V$, $d(v_i, v_j) = 1$ if $(v_i, v_j) \in E$ and $d(v_i, v_j) = 2$ otherwise, $B = |V|$. □

We will now establish that the following problem known as VERTEX COVER is $\mathcal{NP}$-complete and then we will reduce it the HAMILTONIAN CIRCUIT(*directed*) which we will further reduce to HAMILTONIAN CIRCUIT. As a consequence, all these problems, together with TRAVELING SALESMAN are $\mathcal{NP}$-complete.

Problem. VERTEX COVER
Instance: A graph $G = (V, E)$ and a natural number $k \leq |V|$.
Question: Is there some **cover** S in G of size k, i.e., some $S \subset V$ of size k so that every edge of G is incident to some $v \in S$?

Theorem 36. VERTEX COVER *is $\mathcal{NP}$-complete.*

Proof. If S is given to us in the form of a mini oracle then checking that it is indeed a cover of size k requires poly-time. So we are left to show that it is $\mathcal{NP}$-hard. We do this by polynomial-time reducing to it CLIQUE. By HW this suffices.

If $G = (V, E), N$ is an instance of CLIQUE then consider the instance $G' = (V', E'), k$ of VERTEX COVER where $k = |V| - N$, $V' = V$ and $E' = V^2 \setminus E$.

Notice that $S \subseteq V$ is a clique in G if and only if $V \setminus S$ is a vertex cover of G'. To see this:

assume that S is a clique in G and let $(u, v) \in E'$. Then $(u, v) \notin E$ and since S is a clique either u or v are in $V \setminus S$.

conversely assume that S is a not clique in G so there are $u, v \in S$ with $(u, v) \notin E$. So $(u, v) \in E'$ with neither u, v in $V \setminus S$. So $V \setminus S$ is not a cover. □

Theorem 37. VERTEX COVER $\leq_{\mathcal{P}}$ HAMILTONIAN CIRCUIT(*directed graph*).

Proof. Given $G = (V, E)$ and $k \leq V$, we define a directed graph $G' = (V', E')$ as follows. First let $V = \{v_1, \ldots, v_n\}$ and let $e_{ij} = \{v_i, v_j\}$ whenever $(v_i, v_j) \in E$. We will abuse notation and write $e \in E$ for every $e = e_{ij}$ as above. Let V' be the set

$$\{a_1, \ldots, a_k\} \bigcup \{(v, e, 0) \mid v \in V, e \in E, v \in e\} \bigcup \{(v, e, 1) \mid v \in V, e \in E, v \in e\},$$

i.e., or every $e = \{v, w\} \in E$ we include 4 vertexes in G': $(v, e, 0)$, $(v, e, 1)$, $(w, e, 0)$, $(w, e, 1)$. To describe the elements of E' fix any v_i in V and let $v_{j_0}, v_{j_1}, \ldots, v_{j_m}$ be all the neighbors of v_i. Then for every v_i like this and every $l \leq k$ add all arrows in the following path:

$$\begin{aligned} a_l \to (v_i, e_{i,j_0}, 0) \to (v_i, e_{i,j_0}, 1) \to \\ \to (v_i, e_{i,j_1}, 0) \to (v_i, e_{i,j_1}, 1) \to \\ \to \quad \cdots \quad \to \\ \to (v_i, e_{i,j_m}, 0) \to (v_i, e_{i,j_m}, 1) \to a_l \end{aligned}$$

Additionally, for all e_{ij} add $(v_i, e_{i,j}, 0) \longleftrightarrow (v_j, e_{i,j}, 0)$ and $(v_i, e_{i,j}, 1) \longleftrightarrow (v_j, e_{i,j}, 1)$. The construction of G' from G, k can be clearly computed in polynomial time. We are left to show that it is a reduction.

Assume that $v_1, \dots, v_k$ is a vertex cover of G. Consider the path

$$a_1 \to (v_1, e^1_{1,j^1_0}, 0) \to (v_1, e_{1,j^1_0}, 1) \to \cdots \to (v_1, e_{1,j^1_{m_1}}, 0) \to (v_1, e_{1,j^1_{m_1}}, 1) \to$$
$$a_2 \to (v_2, e^2_{1,j^2_0}, 0) \to (v_2, e_{2,j^2_0}, 1) \to \cdots \to (v_2, e_{2,j^2_{m_2}}, 0) \to (v_2, e_{2,j^2_{m_2}}, 1) \to$$
$$\to \quad \cdots \quad \to$$
$$a_k \to (v_k, e^k_{1,j^k_0}, 0) \to (v_k, e_{k,j^k_0}, 1) \to \cdots \to (v_k, e_{k,j^k_{m_k}}, 0) \to (v_k, e_{k,j^k_{m_k}}, 1) \to a_1$$

This path goes over everything exactly once except the vertexes $(v_j, e, 0), (v_j, e, 1)$ with $j > k$. But since $\{v_1, \dots, v_k\}$ is a cover of G, each such e is of the form $e_i j$ for some $i \leq k$. So we can modify the above path replacing $(v_i, e_{ij}, 0) \to (v_i, e_{ij}, 1)$ with

$$(v_i, e_{ij}, 0) \to (v_j, e_{ij}, 0) \to (v_j, e_{ij}, 1) \to (v_i, e_{ij}, 1),$$

wherever appropriate.

Assume now conversely that G' has a Hamiltonian circuit and write it as

$$a_1 \to \cdots \to a_2 \to \cdots\cdots\cdots \to a_k \to \cdots \to a_1$$

The observation now is that for every i there exists some $v_i \in V$ so that every vertex in the path above between a_i, a_{i+1} is either of the form $(v_i, e, 0), (v_i, e, 1)$ or of the form $(w, e, 0), (w, e, 1)$ where $e = \{v_i, w\}$. This is because of the structure of the graph and the way its vertex in V' is stamped with some e and either 0 or 1 (we leave the details to the reader). It follows that $\{v_1, \dots, v_k\}$ is a cover of G. □

Theorem 38. *We have that*

$$\text{HAMILTONIAN CIRCUIT}(\textit{directed graph}) \leq_{\mathcal{P}} \text{HAMILTONIAN CIRCUIT}.$$

Proof. We leave this as a HW. □

4.2. **Some polynomial time problems.** As we pointed out in the HW there is an algorithm solving PRIME in polynomial time. Here we provide with proof a couple of problems which are also solvable in polynomial time. Most importantly the 2-SAT. We start with the following simple problem.

Problem. GRAPH CONNECTEDNESS
Instance: A (possibly directed) graph $G = (V, E)$.
Question: Is G connected?

Theorem 39. GRAPH CONNECTEDNESS *is in* $\mathcal{P}$.

Proof. Given any $u, v \in V$ one can decide whether there is a path from u to v in polynomial time:

Step 1: mark u;
Step $k+1$: if there is an edge from an already marked x to some unmarked y, mark y;
Step $|V|$: stop and answer YES if v is marked.

By induction on the shortest path from u to v we can easily check correctness. It is also clear that the algorithm stops in less than or equal to $|V|^3$ time.

The algorithm for the main problem is to check for all pairs u, v whether there is either a path from u to v or from v to u.

□

Problem. 2-SAT
Instance: as in SAT one is given a finite set V of propositional variables and a finite set C of clauses which are of the form $x^\varepsilon \vee y^\varepsilon$.
Question: is there an assignment of truthvalues $\wedge_{i=1}^m (x_{n_i}^\varepsilon \vee y_{n_i}^\varepsilon)$ true.

Theorem 40. 2-SAT *is in* $\mathcal{P}$.

Proof. It suffice to show that $\neg$(2-SAT) polynomially reduces to a subproblem of GRAPH CONNECTEDNESS.

Let $V = \{v_1, \ldots, v_n\}, C$ be an instance of 2-SAT. We will denote by $x, y, z, \ldots$ the literals of V. If $x = \neg v_i$ then by $\neg x$ we mean v_i. Consider the directed graph $G_C = (V_C, E_C)$ whose vertexes are all literals in V, i.e., V_C consists of:

$$v_1, \ldots, v_n, \neg v_1, \ldots, \neg v_n$$

For any two literals x, y we draw an arrow $x \to y$ if and only if $\neg x \vee y$ is in C. This defines E_C and therefore G_C in a polynomial manner with respect to V, C. The rest follows from the following claim:

Claim. (V, C) *is not in* 2-SAT *if and only if there exists some variable* $v \in V$ *so that within* G_C *there is a path from* v *to* $\neg v$ *and a path from* $\neg v$ *to* v.

The one direction is easy: if in G_C there is a path from x to y, $t\colon V \to \{\top, \bot\}$ is a witness to the satisfiability of C and $t(x) = \top$ then $t(y) = \top$. Since in such t we have that precisely one of $t(v), t(\neg v)$ is $\top$, any path from it to the other would imply that both are $\top$, a contradiction.

We leave the converse as an exercise.

$v \to x$ since that would give a path from $v \to \neg x$ and $\neg x \to \neg\neg v$, i.e., from v to $\neg v$, contradicting the definition process of $t_{k+1}(v)$. Moreover if $t_{k+1}(v) = \top$ and $t_k(z) = \bot$ we cannot have a path from v to z since this would give a path from $\neg z$ to $\neg v$ with $t_k(\neg z) = \top$ and therefore t_k would had been already defined to be $\bot$ on z (similarly for the case where $t_{k+1}(\neg v) = \top$).

□

4.3. **Deciding Presburger arithmetic is exponential-hard.** Recall that by Presburger Arithmetic we mean the complete theory T_{Presb} of $(\mathbb{N}, 0, S, +, <)$. We have seen that T_{Presb} is decidable, i.e., given any sentence σ in $\mathcal{L} = \{0, S, +, <\}$ there is an algorithm deciding whether $\sigma \in T_{\text{Presb}}$, or equivalently, whether

$$(\mathbb{N}, 0, S, +, <) \models \sigma.$$

One could ask whether there is an efficient algorithm deciding T_{Presb}. Here we will show that the problem of deciding T_{Presb} is highly intractable. Recall the complexity class $\mathcal{E} = \bigcup_{c>0} \text{TIME}(2^{cn})$ of all exponential time languages. We have:

Theorem 41. *T_{Presb} is $\mathcal{E}$-hard.*

We devote the rest of this subsection to a proof of this theorem. Let $L \subseteq A^*$ be a language with $L \in \text{TIME}(2^{cn})$ for some $c > 0$. We will describe a polynomial time computable assignment $w \mapsto \sigma_w$ of each $w \in A^*$ to some sentence σ_w so that

$$w \in L \iff \sigma_w \in T_{\text{Presb}}.$$

We view here T_{Presb} as language in a finite alphabet which contains symbols for $\mathcal{L}$ as well as the usual symbols for first order logic. To keep the alphabet finite we code the variable x_n with the finite string xn_{bin} where n_{bin} is the binary expansion of n.

Let $L \in \text{TIME}(2^{cn})$, we can find a Turing machine M on some alphabet $B \supseteq A$ so that M on input w reaches the state Q_Y on time 2^{cn} if and only if $w \in L$. Set $b_0 = \#$, $b_1 = *$, $B = \{b_2, \dots, b_q\}$ and let $\{Q_{q+1}, \dots Q_{p-1}\}$ be the collection of all states of M with $Q_{p-1} = Q_Y$.

Since M terminates in at most 2^{cn} steps we can restrict attention to the interval $[-2^{cn}, 2^{cn}]$ of the tape. In fact it will be convenient to have the head start at the cell 2^{cn} instead and let $[0, 2^{cn+1}]$ be the pertinent interval. Using $b_1 = *$ for the blanks of the tape we can keep track of every situation description by the word:

$$b_{i_0} b_{i_1} \dots b_{i_{k-1}} Q_j b_{i_k} \dots b_{i_{2^{cn}+1}}$$

implying that M is in the state Q_j with the head reading b_{i_k}. Using the letter $b_0 = \#$ as a separator we can represent any computation of length $\leq 2^{cn}$ by another word:

$$x \equiv \#c_0 \# c_1 \# \cdots \# c_{2^{cn}},$$

where every c_i is a situation description as above. Replacing b_i with simply i and Q_j with j we can code the above word x with a single natural number:

$$x = q_0 q_1 \cdots q_t = \Sigma_{j=0}^{t} k_j p^j, \quad \text{where } k_j < p.$$

Here $t + 1 = (2^{cn} + 1) + (2^{cn} + 1)(2^{cn+1} + 2)$ and therefore $x < p^{t+1} < 2^{2^{dn}}$ for some fixed $d > 0$ which only depends on c. Given now $w \in A^*$ with $|w| = n$ we have that:

$$w \in L \iff \exists x \in \mathbb{N}\big(x < 2^{2^{dn}} \text{ and } x \text{ codes a computation with input } w \text{ stopping at } Q_Y\big)$$

$$\iff \exists x P_w(x).$$

Task. Our task now is to find a formula φ_w in $\mathcal{L}$ so that $w \mapsto \varphi_w$ is polynomial and

$$(\mathbb{N}, 0, S, +, <) \models \varphi_w(x) \text{ if and only if } P_w(x) \text{ holds.}$$

This would finish the proof since $w \mapsto \sigma_w := \exists x \varphi_w(x)$ would be the desired reduction showing that $L \leq_{\mathcal{P}} T_{\text{Presb}}$.

We now proceed to show that $P_w(x)$ is indeed definable in $(\mathbb{N}, 0, S, +, <)$. First let $\Phi(x, p, i)$ be the i-th digit of x when x is written in p-ary expansion. We have:

$$P_w(x) \iff \big(x < 2^{2^{dn}}\big) \wedge \big(\forall i \leq 2^{cn}\ \Phi(x, p, i(2^{cn+1}+3)) = 0\big) \wedge$$
$$\wedge \big(\forall i \leq 2^{cn}\ \forall j,\ \text{with } 0 < j \leq 2^{cn+1}+2,\ \ \Phi(x, p, i(2^{cn+1}+3+j)) > 0\big) \wedge$$
$$\wedge \big(\text{describe transitions } c_i \to c_{i+1} \text{ and termination}\big) \wedge$$
$$\wedge \big(\forall k < |w| \Phi(x, p, (2^{cn}+2+k)) = i_k\big),\ \ \text{where } w = b_{i_0} \cdots b_{i_k} \cdots b_{i_{|w|}}.$$

In order to capture $P_w(x)$ with a poly-time computable formula $\varphi_w(x)$ we need to define in $(\mathbb{N}, 0, S, +, <)$ the following expressions in polynomial time from $n = |w|$:

(1) $A_n(x) \quad \leftrightarrow \quad x < 2^n$;
(2) $B_n(x) \quad \leftrightarrow \quad x < 2^{2^n}$;
(3) $M_n(x, y, z) \quad \leftrightarrow \quad (x < 2^{2^n}) \wedge (x \cdot y = z)$;
(4) $E_n(x, y, z) \quad \leftrightarrow \quad (x, z < 2^{2^n}) \wedge (y^x = z)$;
(5) $D_{n,p}(x, y, z) \quad \leftrightarrow \quad (x < 2^{2^n}) \wedge (y < 2^n) \wedge (\Phi(x, p, y) = z)$;

The reader should observe that we are doing a similar coding with the one we did for proving incompleteness in full arithmetic. Of course the theory here is decidable so we cannot do the full Gödel coding. The reason being that we cannot define multiplication and therefore exponentiation and neither the full function $(\Phi(x, p, y) = z)$. For the result we seek to prove though we only need "bounded" versions of these operations which we can define (as we will see) in Presburger arithmetic.

We first point out that it suffice to show that $E_n(x, y, z)$ and $M_n(x, y, z)$ are defined by a poly-time computable formula since:

$$A_n(x) \iff \exists z(E_n(n, 2, z) \wedge x < z), \quad B_n(x) \iff M_n(x, 0, 0), \quad \text{and}$$
$$D_{n,p}(x, y, z) \iff B_n(x) \wedge A_n(y) \wedge \big(\exists q\ \exists r\ (x = q \cdot p^{y+1} + z \cdot p^y + r) \wedge (r < p^y)\big) \iff \ldots$$

We show now that $M_n(x, y, z)$ is definable by a formula in poly-time by induction.

$(n = 0)$**-case.** $M_0(x, y, z) \equiv (x = 0 \wedge z = 0) \vee (x = 1 \wedge z = y)$.

$(n = k + 1)$**-case.** Assume we have defined $M_k(x, y, z)$ in poly-time (we know how to multiply when $x < 2^{2^k}$). We will define $M_{k+1}(x, y, z)$ (we will learn how to multiply when $x < 2^{2^{k+1}}$). If $x < 2^{2^{k+1}}$ then $\sqrt{x} < 2^{2^k}$ and therefore $\lfloor \sqrt{x} \rfloor < 2^{2^k}$. Set $x_1 = \lfloor \sqrt{x} \rfloor$. We have that $x = x_1 + w$ for some w and $x < (x_1 + 1)^2$. Therefore $x_1^2 + w < x_1^2 + 2x_1 + 1$, so $w \leq 2x_1$, i.e., $x = x_1^2 + x_2 + x_3$ where $x_2, x_3 \leq x_1 < 2^{2^k}$. Consequentially we have that $xy = x_1(x_1 y) + x_2 y + x_3 y$ and therefore:

$$M_{k+1}(x, y, z) \iff (x < 2^{2^{k+1}}) \wedge (x \cdot y = z) \iff \exists x_1 x_2 x_3 \exists u \exists w_1 w_2 w_3$$
$$\Big(\big((x = x_1 + x_2 + x_3) \wedge (z = w_1 + w_2 + w_3) \wedge (x_2 \leq x_1) \wedge (x_3 \leq x_1) \wedge (x_1 < 2^{2^k})\big)$$
$$\bigwedge \big((w_2 = x_2 \cdot y) \wedge (w_3 = x_3 \cdot y) \wedge (w_1 = x_1 \cdot u) \wedge (u = x_1 \cdot y)\big) \Big)$$

which can be entirely rewritten using the already defined M_k. However at this point it still seems that the length of the definition of M_{k+1} is ruffly 4-times the length of

the definition of M_k. This would make the computation of M_n exponential. But we can rewrite the above as follows:

$$\iff \exists x_1 x_2 x_3 \exists u \exists w_1 w_2 w_3$$

$$\Big(\big((x = x_1 + x_2 + x_3) \wedge (z = w_1 + w_2 + w_3) \wedge (x_2 \leq x_1) \wedge (x_3 \leq x_1) \wedge (x_1 < 2^{2^k}) \big)$$

$$\bigwedge \big(M_k(x_2, y, w_2) \wedge M_k(x_3, y, w_3) \wedge M_k(x_1, u, w_1) \wedge M_k(x_1, y, u) \big) \Big)$$

$$\iff$$

$$\iff \exists x_1 x_2 x_3 \exists u \exists w_1 w_2 w_3$$

$$\Big(\big((x = x_1 + x_2 + x_3) \wedge (z = w_1 + w_2 + w_3) \wedge (x_2 \leq x_1) \wedge (x_3 \leq x_1) \wedge (x_1 < 2^{2^k}) \big)$$

$$\bigwedge \forall r, s, t \big((r, s, t) \in \{(x_2, y, w_2), (x_3, y, w_3), (x_1, u, w_1), (x_1, y, u)\} \implies M_k(r, s, t) \big) \Big)$$

The length of this formula M_n has now polynomial dependence on n. However it seems that going from k to $k+1$ we are forced to introduce new variables r, s, t each time so the length of the expressions still grows rapidly (but at least polynomially rapidly). However there is a way to define $M_k(x, y, z)$ using uniformly the following **fixed** collection of variables: $x, y, z, x_1, x_2, x_3, u, w_1, w_2, w_3, r', s', t'$. This makes the rate of growth linear!

5. A closer look on complexity

In the previous section we showed that the decision problem of Presburger arithmetic is $\mathcal{E}$-hard. However this does not imply, apriori, that the problem is intractable. For all we know the collections $\mathcal{E}$ and $\mathcal{P}$ make contain precisely the same languages. Here we establish that this is not the case. Then we place our attention to another notion of complexity which measures the space, rather than the time, needed for a certain problem to be decided. We finally show that, for Turing machines which have access to oracles, whether $\mathcal{P} \neq \mathcal{NP}$ or $\mathcal{P} = \mathcal{NP}$ holds depends on the oracle.

It will be convenient to change slightly the model of computation we use by allowing the Turing machine to have multiple tapes. We will leave it to the reader to check that the translation between single tape and multiple tape model introduces at most quadratic slow down in time and therefore it will not affect the coarse complexity bound we are interested in.

A **k-string Turing machine** M ($k \geq 1$) is defined as a usual Turing machine by specifying its alphabet Σ, its collection of states $Q_0, \ldots, Q_p$ (with Q_0 initial and Q_p terminal) and its transition function δ where

$$\delta \colon \{Q_0, \ldots, Q_p\} \times \Sigma^k \quad \to \quad \Sigma^k \times \{\text{left}, \text{right}, \text{stay}\}^k \times \{Q_0, \ldots, Q_p\}.$$

In other words, there are k-many heads each reading some cell of some of the k-tapes. We will assume always that the **input** is placed at the first (the 0th) tape and the **output** at the last (the $(k-1)$th) tape

Given a function $f \colon \mathbb{N} \to \mathbb{N}$ we say that a computation halts in $\mathcal{O}(f(n))$ if there are $a, b > 0$ so that the computation halts in less than $a \cdot f(n) + b$ time.

Exercise 42. *Given a k-string Turing machine M which always halts in less than $f(n)$ time describe a Turing machine M' which always halts with the same output as M on the same input but in time $\mathcal{O}(f^2(n))$*

5.1. **The hierarchy theorem.** In this subsection the time complexity is always measured with respect to a k-string Turing machines. In other words, a language $\Sigma \subseteq A^*$ belongs to some time complexity class if and only if there is some k and some k-string Turing machine M that decides this problem within the specified time constrains.

Let $f \colon \mathbb{N} \to \mathbb{N}$ be any map. We say that f is a **proper complexity function** if $f(n) \geq n$, $f(n)$ is non-decreasing, and there exists some k-string Turing machine on alphabet Σ with $1 \in \Sigma$ so that for any input x with $|x| = n$ M halts in $\text{TIME}(f(n))$ time with output

$$\overbrace{11\cdots11}^{n-\text{times}}$$

Theorem 43 (Time Hierarchy Theorem)**.** *If f is a proper complexity function then there exists a language L whose decision problem is in* $\text{TIME}(\big(f(2n+1)\big)^3)$ *but not in* $\text{TIME}(f(n))$.

Before we proceed to the proof we point out the following immediate corollary.

Corollary 44. *The class $\mathcal{E}$ is strictly larger than $\mathcal{P}$, and therefore, the decision problem for the theory of Presburger arithmetic is intractable.*

Proof of Corollary. In the context of Exercise 42 it follows immediately from the Time Hierarchy Theorem since 2^n is easily seen to be a proper complexity function. □

We now turn into the proof of the Time Hierarchy theorem. Let f be a proper complexity function and consider the following bounded version of the halting problem which we view as a language H_f in the alphabet $A = \{0, 1, (,), ;\}$:

$$H_f = \{M; x \mid M \text{ is a } k\text{-string TM which accepts } x \text{ in at most } f(|x|) \text{ steps}\}.$$

One can code the states and the alphabet of the Turing machine M with natural numbers in binary (for each M use strings of $0, 1$ of some fixed large enough length in order to code states and alphabet). Similarly the transition function can be coded using tuples (use parenthesis and ; instead of comma) so the alphabet A is enough for the coding.

The Time Hierarchy theorem follows from combining the next two lemmas under the obvious re-parametrization.

Lemma 45. *The decision problem for H_f is in* $\mathrm{TIME}(\big(f(n)\big)^3)$.

Proof. HW. □

Lemma 46. *The decision problem for H_f is not in* $\mathrm{TIME}(f(\lfloor \frac{n}{2} \rfloor))$.

Proof. Assume towards contradiction that there is some k-string Turing machine N which on input $M; x$ decides H_f in $\mathrm{TIME}(f(\lfloor \frac{n}{2} \rfloor))$ consider the usual antidiagonalizing machine D which takes as input descriptions of Turing machines M, then calls N with input $M; M$, and then accepts M if and only if N rejects $M; M$. Notice that D, on input M, runs in the same time as N on input $M; M$. That is if $|M| = n$ then D needs $f(\lfloor \frac{2n+1}{2} \rfloor) = f(n)$.

As usual we come into contradiction by asking whether $D; D \in H_f$: if the answer is yes then D accepts D in $f(n)$; i.e., N rejects $D; D$ in $f(\lfloor \frac{n}{2} \rfloor) \leq f(n)$, and therefore $D; D \notin H_f$. Similarly for the other direction. □

5.2. **Space and non-deterministic space complexity.** This k-string Turing machine model allows us to measure complexity in terms of another resource, i.e., space. How many cells of tape are necessary for deciding a language. When counting the number of cells we want to exclude the cells used to write the input and the output. We do that as follows. A **restricted k-string Turing machine** is just a k-string Turing machine with $k \geq 2$ so that the input tape (0th) is read only and the output tape ($(k-1)$th) is write only. That is, on the first tape the transition function δ can never replace any symbol with another, and in the last tape, δ can order the

head to go only to $\{\text{right}, \text{stay}\}$ excluding "left". We call the tapes $1, 2, \ldots, k-2$ the **work tapes**.

We say that some language $L \subseteq A^*$ is **of space complexity** $f(n)$ and write $L \in \text{SPACE}(f(n))$ if there is a restricted k-string Turing machine M so that for every $x \in A^*$ there are $f(|x|)$-many cells of the work tape (which we call **free work space**) so that accepts x after a computation which only used the free work space from the work tapes if $x \in L$; and M rejects x after a computation which only used the free work space from the work tapes if $x \notin L$. Here we will care about the class $\mathcal{P}\text{SPACE}$ of all languages which can be decided in $\text{SPACE}(p(n))$ for some polynomial p.

One can also define the non-deterministic analogue of SPACE complexity (although we will not use it): introduce another tape, say the tape (-1), on which an arbitrary long finite "verifier" (or "mini-oracle") can be placed and impose two restrictions on this tape so that we cannot use it to "cheat" with respect to space. The first restriction is that this tape, as in the case of the 0th tape, is read only. The second restriction is that, as in the case of the $(k-1)$th tape, the head each time moves only $\{\text{right}, \text{stay}\}$ and never "left". So as the computations proceeds we cannot recover the history of the tape except the part that we copy on the work tape (which requires from us to have more free work space in the work tapes). Call such a Turing machine **non-deterministic restricted k-string Turing machine**.

A language $L \subseteq A^*$ is in $\mathcal{NP}\text{SPACE}$ if there is non-deterministic restricted k-string Turing machine M in some alphabet $\Sigma \supseteq A$ and a polynomial $p(n)$ so that for every $x \in A^*$ there is a "mini-oracle" $y \in \Sigma^*$ so that M accepts (x, y) using at most $p(|x|)$ free work space (from tapes $1, 2, \ldots, k-2$), if and only if $x \in L$.

Interestingly one can prove the following theorem.

Theorem 47. $\mathcal{P}\text{SPACE} = \mathcal{NP}\text{SPACE}$

The proof of this theorem goes through the next lemma. Recall the following problem which implicitly appeared in the proof of Theorem 39.

Problem. REACH(G, s, t)
Instance: A directed graph $G = (V, E)$ together with $s, t \in V$.
Question: Is there a path from s to t?

The procedure we described there uses, worst case, $|V|$ amount of space to work because it has to keep track of the "marked" vertexes.

Theorem 48. *Let $g\colon \mathbb{N} \to \mathbb{N}$ be a function with $g(n) \geq \log(n)$. Assume we are given $G = (V, E)$, $s, t \in V$ and $l \in \mathbb{N}$ so that deciding "is there an edge between $v, w \in V$" can be achieved in $\mathcal{O}(g(n))$ space (where n is the length of the coding of G, S, t, l). Then we can decide "is there a path between s, t of length at most l" can be achieved in $\mathcal{O}\big(g(n)\log(l)\big)$ space.*

We will use the fact that there is a language L that is $\mathcal{P}$SPACE-complete.

Problem. HALT WITHOUT EXPANSION(M, x)
Instance: A (usual single-tape) Turing machine M and an input x.
Question: Does M accepts x by a computation which uses no more tape then the cells that were initially occupied by x?

Theorem 49. HALT WITHOUT EXPANSION(M, x) *is* $\mathcal{P}$SPACE*-complete.*

Proof. HW □

5.3. $\mathcal{P} =^? \mathcal{NP}$ **depends on oracles.** In this subsection we assume that all languages are given in the alphabet $A = \{0, 1\}$. This will not impose any restrictions since any other language in any finite alphabet B reduces in linear time to a language in A. By an oracle R we mean any subset of A^*. One can of course fix an effective enumeration of A^* with $\mathbb{N}$ and identify oracles R, in the above sense, with $\alpha \in \mathbb{N}^{\mathbb{N}}$. However, in this context it is more natural to work directly with R.

A usual (or a k-string Turing machine/restricted k-string Turing machine) Turing machine $M = M[\cdot]$ is an **oracle machine** if it has an additional tape, the **query tape** and a special state $Q_?$ which functions as follows: first we attach M to some oracle R to get $M[R]$; then $M[R]$ runs according the usual instructions; if at some point the state $Q_?$ is reached the we ask the oracle if the string written in the query tape is in R, if yes then string in the query tape gets replaced by a single 1, if not by a single 0. This latter operation of asking the oracle costs a single unit of time.

Convention. Whenever we write a property about $M[]$ without specifying an oracle we mean that this property holds for all possible oracles.

In the presence of some fixed oracle R we can define the various complexity classes such as $\mathcal{P}$ and $\mathcal{NP}$ as follows:

• $\mathcal{P}^R$ is the collection of all languages $L \subseteq A^*$ for which there exists an oracle machine $M[]$ and a polynomial p so that for all $x \in A^*$, $M[R]$ halts for input x in at most time $p(|x|)$ and it accepts x if and only if $x \in L$.

• $\mathcal{NP}^R$ is the collection of all languages $L \subseteq A^*$ for which there exists an oracle machine $M[]$ and a polynomial p so that for all $x \in A^*$ and every $y \in A^*$, $M[R]$ halts for input $x\#y$ in at most time $p(|x|)$ time and

$$x \in L \iff \exists y \in A^* \ |y| < p(|x|) \text{ so that } M[R] \text{ accepts } x\#y.$$

Theorem 50. *There are oracles $S, T \subseteq A^*$ so that:*

$$\mathcal{P}^S = \mathcal{NP}^S \quad \text{and} \quad \mathcal{P}^T \neq \mathcal{NP}^T.$$

We start by proving the first part of this theorem. For S take any language that is $\mathcal{P}$SPACE-complete. For example one can take the language of Theorem 49 (coded in binary which can always be done in linear time). Since for every oracle R we have that $\mathcal{P}^R = \mathcal{NP}^R$, it suffices to show that $\mathcal{NP}^S \subseteq \mathcal{P}^S$. To see this notice that

(**Claim:**) $$\mathcal{NP}^S \subseteq \mathcal{P}\text{SPACE} \quad \text{and} \quad \mathcal{P}\text{SPACE} \subseteq \mathcal{P}^S$$

Proof of Claim. Let $L \subseteq A^*$ be in $\mathcal{NP}^S$ and let $M[]$ be the oracle machine and $p(n)$ the polynomial with $x \in L$ if and only if $\exists y \in A^*\ |y| < p(|x|)$ so that $M[S]$ accepts $x\#y$ in $p(|x|)$ time. Consider now the restricted 5-tape Turing machine N which has two work tapes besides $1, 2, 3$ besides the input tape 0 and the output tape 4. Given the input x the machine N produces the first $y = y_1 \in A^{<p(n)}$ (chose here any enumeration) in tape 1. Then it uses tape 2 to imitate the computation of $M[]$. If $M[]$ asks at some point the oracle S about a certain string w then N uses the tape 3 to solve whether $w \in S$.

Each such y_1 takes $p(|x|)$ space in tape 1, $p(|x|)$ time and therefore $p(|x|)$ space in tape 2 and $p'(|x|)$ space to solve each of the polynomialy many questions of the form $w \in S$ in tape 3, where $p'(n)$ depends on $p(n)$ and the polynomial $q(n)$ which bounds the space complexity of solving S on a single tape machine. If y_1 rejects reuse the same space to try y_2. Total space needed: $2p(n) + p'(n)$. So we have:

$$\mathcal{NP}^S \subseteq \mathcal{P}\text{SPACE}$$

The other inclusion is easy: if $L \subseteq A^*$ is an problem in $\mathcal{P}$SPACE then there is a polynomial time reduction $f\colon A^* \to A^*$ of L to S since S is $\mathcal{P}$SPACE-complete. Let N_f be the Turing machine that computes this reduction and turn in into an oracle machine $N'_f[]$ which after the reduction asks the oracle about $f(x)$. Then $N'_f[S]$ decides L in polynomial time. □

This finishes the first part of the proof of Theorem 50. Out next task is to find an oracle T so that $\mathcal{P}^T \neq \mathcal{NP}^T$.

First to each oracle $R \subseteq A^*$ we associate the language L_R with

$$L_R = \{1^n \mid \exists w \in R\ |w| = n\} = \{1^n \mid R \cap A^n \neq \emptyset\}.$$

We will construct some T for which $L_T \in \mathcal{NP}^T \setminus \mathcal{P}^T$. It is immediate that for every T the language L_T is in $\mathcal{NP}^T$. So just need to find some T with the property that $L_T \notin \mathcal{P}^T$. Next lemma is a warm-up:

Lemma 51. *Let $M[\cdot]$ be an oracle machine which halts in polynomial time. Then there exists some oracle $R \subseteq A^*$ so that L_R is not decided by $M[R]$.*

Proof. Let p be the polynomial which bounds the halting time of M and let k be the first natural number with $2^k > p(k)$. We will define R so that

$$M[R] \text{ accepts } 1^k \quad \Longleftrightarrow \quad R \cap A^n = \emptyset.$$

We define R by specifying for every $w \in A^*$ whether $w \in R$ as follows:

(1) if $k < |w|$ chose either $w \in R$ or $w \notin R$, it doesn't matter;
(2) if $k < |w| \leq p(k)$ chose either $w \in R$ or $w \notin R$, again it doesn't matter;
(3) for $|w| = k$: having fixed choices for (1), (2) above we run $M[\cdot]$ with oracle the partially defined R and input 1^k. If at some point during the computation $M[\cdot]$ asks whether $w \in R$ where $|w| = k$ then we extend the definition of R so that $w \notin R$. We keep on doing this for all questions "$w \in R$?" where $|w| = k$ we encounter. Since $M[\cdot]$ halts in $p(k)$ time it will never be able to

ask questions "$w \in R$?" where $|w| > p(k)$. Hence info we provided in (1), (2), (3) is enough for the computation of $M[R]$ to halt for input 1^k. But now notice that since there are at most $p(k)$ many questions "$w \in R$?" and $2^k > k$ there is at least some $w_R \in A^k$ for which M did not ask whether "$w_R \in R$". It $M[R]$ accepted 1^k then put $w \notin R$ for all $w \in A^k$. Otherwise if $M[R]$ rejected 1^k put $w_R \in R$;

(4) if $|w| > p(k)$ as in (1), (2) do whatever you want. We separated this case to emphasize (for later) that the choices of step (3) do not depend on step (4)

This concludes the proof of the lemma. □

Exercise 52. *Finish the proof of "there is T such that $\mathcal{P}^T \neq \mathcal{NP}^T$". If you prefer try showing something stronger. Namely, that for the generic oracle $T \subseteq \{0,1\}^*$ we have $\mathcal{P}^T \neq \mathcal{NP}^T$ (see HW 6).*

6. A combinatorial sentence not provable from Peano arithmetic

We will prove that a certain true combinatorial statement known as the Paris-Harrington principle is not provable from Peano arithmetic. Before we get into the proof of this negative result we will review some finite and infinite combinatorics and provide an application.

6.1. **The Ramsey theorem and the Paris-Harrington principle.** In what follows we will often identify n with the set $\{0, \dots, n-1\}$ of all its predecessors. If X is any set and $k \in \mathbb{N}$ then we denote by $[X]^k$ the set of all **k-sets of X**, i.e. all subsets of X of size k. Recall Ramsey's theorem.

Theorem 53 (Finite Ramsey theorem)**.** *For every $a, b \in \mathbb{N}$ and every $r \in \mathbb{N}$ there is $c \in \mathbb{N}$ so that*

$$c \to (b)^a_r,$$

where $c \to (b)^a_r$ stands for:

"for every $f \colon [c]^a \to r$ there is $X \subseteq c$ with $|X| \geq b$ so that f is constant on $[X]^a$"

The Paris-Harrington principle is a slight strengthening of Ramsey's theorem.

Theorem 54 (Paris-Harrington principle)**.** *For every $a, b \in \mathbb{N}$ and every $r \in \mathbb{N}$ there is $c \in \mathbb{N}$ so that*

$$c \to (b)^a_r,$$

where $c \to_* (b)^a_r$ stands for:

"for every $f \colon [c]^a \to_* r$ there is $X \subseteq c$ with $|X| \geq b$ and $|X| \geq \min X$, so that f is constant on $[X]^a$"

Both statements follow from the infinite Ramsey theorem which is a theorem of set theory. Notice that the infinite Ramsey theorem cannot be stated in first order arithmetic. It is a homework exercise to show that both the finite Ramsey statement and the Paris-Harrington principle can be expressed in first order arithmetic.

Theorem 55 (Infinite Ramsey theorem)**.** *For every $a \in \mathbb{N}$ and every $r \in \mathbb{N}$ we have*

$$\mathbb{N} \to (\mathbb{N})^a_r,$$

where $\mathbb{N} \to (\mathbb{N})^a_r$ stands for:

"for every $f \colon [\mathbb{N}]^a \to r$ there is an infinite $X \subseteq c$ so that f is constant on $[X]^a$"

Proof. To be added. □

Proof of Theorems 53 and 54 from Theorem 55. To be added. □

6.2. **Indiscernibles.** Ramsey type of results allow us to some hidden structure within model theoretic structures.

Let $\mathcal{M}$ be a structure and let I be a set indexing some collection $\{a_i \mid i \in I\}$ of elements of M. We say that $(a_i)_{i\in I}$ is an **indiscernible sequence** in $\mathcal{M}$ if for every formula $\varphi(x_1, \ldots, x_m)$ and every $i_1, \ldots, i_m, j_1, \ldots, j_m \in I$ we have that

$$\mathcal{M} \models \varphi(a_{i_1}, \ldots, a_{i_m}) \iff \varphi(a_{j_1}, \ldots, a_{j_m}).$$

Any basis in a vector space forms an indiscernible sequence. However there are theories T which have no models containing infinite sequences of indiscernibles (with $a_i \neq a_j$). Such is the theory of linear orders since $a_i < a_j \implies a_j \not< a_i$. Interestingly that is the only obstruction to having indiscernibles.

Let $\mathcal{M}$ be a structure and let $(I, <)$ be an ordered set indexing some collection $\{a_i \mid i \in I\}$ of elements of M. We say that $(a_i)_{i\in I}$ is a **sequence of order indiscernibles** in $\mathcal{M}$ if for every formula $\varphi(x_1, \ldots, x_m)$ and every $i_1 < \ldots < i_m$, $j_1 < \ldots < j_m \in I$ we have that

$$\mathcal{M} \models \varphi(a_{i_1}, \ldots, a_{i_m}) \iff \varphi(a_{j_1}, \ldots, a_{j_m}).$$

Theorem 56. *Let T be a theory having infinite models. Then, for any linearly ordered set $(I, <)$ there is $\mathcal{M} \models T$ and a sequence $(a_i)_{i\in I}$ in M of order indiscernibles.*

Proof. Let $\mathcal{L}' = \mathcal{L} \cup \{c_i \mid i \in I\}$ and consider the theory T' containing:

(1) every axiom in T;
(2) $c_i \neq c_j$ for all $i \neq j$;
(3) $\varphi(a_{i_1}, \ldots, a_{i_m}) \iff \varphi(a_{j_1}, \ldots, a_{j_m})$ for every formula and all $i_1 < \ldots < i_m$, $j_1 < \ldots < j_m \in I$.

The $\mathcal{L}$-reduct $\mathcal{M}$ of any model $\mathcal{M}'$ of T' satisfies the conclusion of the theorem. Hence we need to show that T' is satisfiable. By compactness, it suffices to show that every finite subset of T is satisfiable.

Let S be a finite subset of T' and let J be a finite subset of I so that if c_j appears somewhere in S then $j \in J$. Let also $\{\varphi_1, \ldots, \varphi_n\}$ be the subset of T' containing all formulas which appear in the axioms of type (3). We will assume without loss of generality that all these φ_i are in the same set of variables $x_1, \ldots, x_m$ (one can insert dummy variables if necessary).

Let now $\mathcal{N}$ be any infinite model of T and fix any linear order $<$ on N. Consider the the coloring $r\colon [N]^m \to 2^n$ with

$$r(a_{i_1} < \ldots < a_{i_m}) = \chi_A,$$

where $A = \{k < n \mid \mathcal{N} \models \varphi_k(a_{i_1}, \ldots, a_{i_m})\}$. By the infinite Ramsey theorem (the finite would also suffice) we get an infinite subset L of N so that r is constant on $[L]^m$. Pick $|J|$ many elements of L and label them by the $(c_j)_{j\in J}$. This produces a structure $\mathcal{N}'$ in the language $\mathcal{L} \cup \{c_j \mid j \in J\}$ with $\mathcal{N}' \models S$, showing that T is indeed finitely satisfiable. □

To prove the that the Paris-Harrington principle is independent of Peano arithmetic we will need to show that it implies existence of a certain stronger form of

indiscernibles for models of Peano arithmetic. First we need to show that it implies another combinatorial principle.

Definition 57. Let $X \subseteq \mathbb{N}$ and let $f\colon [X]^n \to \mathbb{N}$ be any map. We say that f is **regressive** if for all $A \in [X]^n$ we have that $f(A) < \min A$. A subset Y of X is said to be **min-homogeneous for** f if for every $A, B \in [X]^n$ with $\min(A) = \min(B)$ we have that $f(A) = f(B)$.

If $s < t$ in $\mathbb{N}$ we will denote by $[s,t]$ and by (s,t) the sets $\{s, s+1, \dots, t-1, t\}$ and $\{s+1, \dots, t-1\}$ respectively. Consider the following combinatorial principle which we will refer to by principle $(\star)$.

$(\star)$ for all $a, b, s, k \in \mathbb{N}$ there is $c \in \mathbb{N}$ so that if $f_1, \dots, f_k\colon [c]^a \to c$ are regressive, then there is $Y \subseteq (s,c)$ with $|Y| \geq b$ and Y is min-homogeneous for all f_i.

We will prove that the Paris-Harrington principle implies principle $(\star)$. Moreover, it will be important that our proof for "Paris-Harrington principle implies principle $(\star)$" will be formalizable in Peano arithmetic.

Lemma 58. *Assume the Paris-Harrington principle. Then for all $a', b', s', r' \in \mathbb{N}$ there is $c' \in \mathbb{N}$ so that for all $g\colon [c']^{a'} \to r'$ there is $Y \subseteq (s', c')$ with $|Y| \geq b' + a'$, $|Y| \geq \min Y + a' + 1$ and g is constant on $[Y]^{a'}$.*

Proof. We could try to use Paris-Harrington for the same a, r and for $b = b' + a' + s'$. The set Y we would get, would satisfy $Y \subseteq (s', c')$ and $|Y| \geq b' + a'$ but not necessarily $|Y| \geq \min Y + a' + 1$ in general. We need a further trick.

Let c be the number given by Paris-Harrington for a, b, r with $a = a', r = r' + 1$ and $b = b' + 2a' + s' + 1$. So for every $f\colon [c]^a \to r' + 1$ there is Z with $Z \geq b' + 2a' + s' + 1, \min Z$ so that $f \restriction [Z]^a$ is constant.

We claim that setting c' to be this c does the job. To see this let $g\colon [c']^{a'} \to r'$ be any coloring and consider a new coloring $f\colon [c']^{a'} \to r'+1$ so that $f(\{y_1, \dots, y_a\}) = r'$ if $y_i < a' + s' + 1$ for some i, and $f(\{y_1, \dots, y_a\}) = g(\{y_1 - a - 1, \dots, y_a - a - 1\})$ otherwise. We get Z so that $f \restriction [Z]^a$ is constant and $Z \geq b'+2a'+s'+1, \min Z$. Since $Z \geq b'+2a'+s'+1$ there are some $y_1, \dots, y_a \geq a'+s'+1$ so the $f \restriction [Z]^a$ is not constantly equal to r' but constantly equal to something less than r'. As a consequence all $y \in Z$ have the property that $y > a' + s' + 1$ so letting $Y = \{z - a' - 1 \mid z \in Z\}$ we get the desired set: $|Y| \geq b'+2a'+s'+1, Y \subseteq (s', c)$ and $|Y| = |Z| \geq \min Z \geq \min Y + a' + 1$ and $g \restriction [Y]^{a'}$ is constant. □

Lemma 59. *The previous lemma implies the $(\star)$-principle. That is, for all $a, b, s, k \in \mathbb{N}$ there is $c \in \mathbb{N}$ so that if $f_1, \dots, f_k\colon [c]^a \to c$ are regressive, then there is $X \subseteq [s, c)$ with $|X| \geq b$ and X is min-homogeneous for all f_i.*

Proof. Let $c = c'$ where c' is attained from the previous lemma by setting $a' = a + 1, s' = s, b' = b$ and $r' = 3^k$. For the rest of the proof we just check that this c does the job we want.

Assume that $f_1, \dots, f_k \colon [c]^a \to c$ are regressive and define $g_1, \dots, g_k \colon [c]^{a+1} \to c$ as follows: for every i and every $A = \{a_0 < a_1 < a_2 < \dots < a_{n-1} < a_n\}$

$$g_i(A) = 0 \iff f_i(\{a_0, a_1, \dots, a_{n-1}\}) = f_i(\{a_0, a_2, \dots, a_n\})$$
$$g_i(A) = 1 \iff f_i(\{a_0, a_1, \dots, a_{n-1}\}) < f_i(\{a_0, a_2, \dots, a_n\})$$
$$g_i(A) = 2 \iff f_i(\{a_0, a_1, \dots, a_{n-1}\}) > f_i(\{a_0, a_2, \dots, a_n\})$$

Combining them we have a single $g \colon [c]^{a+1} \to 3^k$ and therefore from the previous lemma we get $Y \subseteq (s, c)$ with $g \restriction [Y]^{a+1}$ constant and $|Y| \geq \min Y + a + 1, a + b$

Claim. $g_i(A) = 0$ for all $A \in [Y]^{a+1}$ and all g_i.

To see this let $\{y_0 < y_1 < \dots < y_l\}$ be an enumeration of Y. For all $j \in \{1, \dots, l - a + 1\}$ let $\bar{y}_j = (y_j, y_{j+1}, \dots, y_{j+a-1})$. Since f_i is regressive, $f_i(y_0 \bar{y}_j) < y_0$.

So we have $l - a + 1$ many sets of the form $y_0 \bar{y}_j$ and only y_0 many slots for $f_i(y_0 \bar{y}_j)$, where $y_0 = \min Y \leq |Y| - a - 1 = l + 1 - a - 1 = l - a$. By pigeonhole principle there are two tuples $y_0 \bar{y}_j, y_0 \bar{y}_{j'}$ so that $f_i(y_0 \bar{y}_j) = f_i(y_0 \bar{y}_{j'})$ and since g_i is constant on $A \in [Y]^{a+1}$ we have that its range is equal to 0 on $[Y]^{a+1}$.

Let now $\{z_1 < \dots < z_{a-1}\}$ the largest $a - 1$ elements of Y and set $X = Y \setminus \{z_1, \dots, z_{a-1}\}$. We claim that this is the desired set X. Clearly we have that $|X| > b$ and $Y \subseteq (s, c)$. So it suffices to show that X is min-homogeneous for every f_i. Let $x_1 < \dots < x_a$ and $x'_1 < \dots < x'_a$ be two a-sets in X with $x_1 = x'_1$. We have that

$$f_i(x_1, x_2, \dots, x_a) = f_i(x_1, x_3, \dots, x_a, z_1) = \dots = f_i(x_1, z_1, \dots, z_{a-1})$$

and similarly

$$f_i(x'_1, x'_2, \dots, x'_a) = f_i(x'_1, x'_3, \dots, x'_a, z_1) = \dots = f_i(x'_1, z_1, \dots, z_{a-1}).$$

Where the later are equal since $x_1 = x'_1$.

□

Let $\mathcal{M}$ be a model of Peano arithmetic and let Φ be a set of formulas of the form $\varphi(x_1, \ldots, x_k, y_1, \ldots, y_n)$. A subset I of M is said to be a **sequence of diagonal indiscernibles** for Φ if for every $\varphi \in \Phi$ and every $a, b_1, \ldots, b_n, c_1, \ldots, c_n \in I$ with $a < b_1 < \cdots < b_n$ and $a < c_1 < \cdots < c_n$ and any $a_1, \ldots, a_k < a$ we have that:

$$\mathcal{M} \models \varphi(\bar{a}, \bar{b}) \iff \mathcal{M} \models \varphi(\bar{a}, \bar{c}).$$

The principle $(\star)$ above allows us to produce arbitrary long sequences of diagonal indiscernibles with respect to any finite Φ in the standard model of arithmetic.

Lemma 60. *For any $k, l, m, n \in \mathbb{N}$ and any set of formulas $\varphi_1(x_1, \ldots, x_k, y_1, \ldots, y_n)$, $\ldots, \varphi_l(x_1, \ldots, x_k, y_1, \ldots, y_n)$ in the language of arithmetic, there is $I \subseteq \mathbb{N}$ of diagonal indiscernibles for $\{\varphi_1, \ldots, \varphi_l\}$ with $|I| \geq m$.*

Proof. We may assume that $m > 2n$. We will find a large enough $s \in \mathbb{N}$ so within $[0, s]$ there will be a large set X with almost the property we need, namely, for every $b_0 < b_1 < \ldots < b_n < b_{n+1} < \ldots < b_{2n} \in X$ and any $a_1, \ldots, a_k < b_0$ we will have $\mathcal{M} \models \varphi_i(\bar{a}, b_0, \ldots, b_n) \iff \mathcal{M} \models \varphi_i(\bar{a}, b_{n+1}, \ldots, b_{2n})$ for all $i \leq l$.

To motivate the definition of s consider the any such $B = \{b_0, \ldots, b_{2n}\} \subseteq [0, s]$ of size $2n + 1$. If the above property fails, then there is some witness to this failure, i.e., some $a_1, \ldots, a_k < b_0$ and some $i \in \{1, \ldots, l\}$. We can define therefore functions $f_1, \ldots, f_k \colon [s]^{2n+1} \to s$ and $g \colon [s]^{2n+1} \to \{0, \ldots, l\}$ so that, if B fails the above property $f_1(B) = a_1, \ldots, f_k(B) = a_k$, $g(B) = i$ is any witness to this failure and if B satisfies the above property then $f_1(B) = \ldots = f_k(B) = g(B) = 0$. Notice that the functions $f_1, \ldots, f_k$ are regressive!

By the finite Ramsey theorem we have some $w \in \mathbb{N}$ so that $w \to (m + n)^{2n+1}_{l+1}$. By principle $(\star)$ we can then find s so that whenever $f_1, \ldots, f_k \colon [s]^{2n+1} \to s$ are regressive, there is $Y \subseteq (w, s)$ with $|Y| = w$ so that Y is min-homogeneous for all f_j. As a consequence, for the above defined $f_1, \ldots, f_n$ we find such a min-homogenous set Y of size w. Since $|Y| = w$ and $w \to (m+n)^{2n+1}_{l+1}$ we can restrict further to some $X \subseteq Y$ with $|X| = m + n$ so that the g defined above is constant on $[X]^{2n+1}$.

Claim. $g(B) = 0$ and therefore $f_1(B) = \cdots = f_n(B) = 0$ for all $B \in [X]^{2n+1}$.

Proof. Assume that $g(B) = i > 0$. Notice that since $m > 2n$ we can find a sequence of length $3n + 1$ in X:

$$b_0 \quad < \quad b_1 < \cdots < b_n \quad < \quad b_{n+1} < \cdots < b_{2n} \quad < \quad b_{2n+1} < \cdots < b_{3n}.$$

Let $\bar{a} = (f_1(B), \ldots, f_k(B))$ for any $B[X]^{2n+1}$ with $\min B = b_0$. We then have that

$$\varphi_i(\bar{a}, b_1, \ldots, b_n) \not\leftrightarrow \varphi_i(\bar{a}, b_{n+1}, \ldots, b_{2n}),$$

$$\varphi_i(\bar{a}, b_{n+1}, \ldots, b_{2n}) \not\leftrightarrow \varphi_i(\bar{a}, b_{2n+1}, \ldots, b_{3n}), \text{and}$$

$$\varphi_i(\bar{a}, b_1, \ldots, b_n) \not\leftrightarrow \varphi_i(\bar{a}, b_{2n+1}, \ldots, b_{3n})$$

Contradiction: since at least two of the above ought to have the same truthvalue.

To finish with the proof we set $d_1 < \ldots < d_n$ to be the n many largest elements of X and set $I = X \setminus \{d_1, \ldots, d_n\}$. We have $|I| = m$ and for every

$a, b_1, \dots, b_n, c_1, \dots, c_n \in I$ with $a < b_1 < \cdots < b_n$ and $a < c_1 < \cdots < c_n$ and any $a_1, \dots, a_k < a$ we have that:

$$\mathcal{M} \models \varphi(\bar{a}, \bar{b}) \iff \mathcal{M} \models \varphi(\bar{a}, \bar{d}) \iff \mathcal{M} \models \varphi(\bar{a}, \bar{c}).$$

Since $a < b_1 < \cdots < b_n < d_1 < \cdots < d_n$ and $a < c_1 < \cdots < c_n < d_1 < \cdots < d_n$ are in X

□

One may use infinite sequences of diagonal indiscernibles to construct "inner models" of Peano arithmetic. Recall from HW that a Δ_0-formula is a formula that is built using only with bounded quantification.

Lemma 61. *Let $\mathcal{M}$ be a model of Peano arithmetic and let $b_0 < b_1 < \cdots$ be an infinite sequence of diagonal indiscernibles for all Δ_0-formulas. Let $N \subseteq M$ be the set $\{a \in M \mid \exists n\ a < b_n\}$ and let $\mathcal{N}$. Then*

(1) N is closed under addition and multiplication;
(2) $\mathcal{N} := \mathcal{M}\restriction N$ is a model of Peano arithmetic.

Proof. **Addition.** Let $b < b' < b'' < b'''$ be in the sequence of indiscernibles and let $a < b$ be any element. It suffice to show that $a + b' < b''$. If not then there is some $a' < a$ so that $a' + b' = b''$. But then, by diagonal indiscernibility we have that $a' + b' = b'''$ as well which contradicts that $b'' < b'''$.

Multiplication. Similar and we leave it as an exercise.

$\mathcal{N}$ **models PA.** This follows from the HW since the validity of an arbitrary formula in $\mathcal{N}$ reduces to the validity of a Δ_0-formula in $\mathcal{M}$.

For example, consider the formula $\varphi(\bar{x}) \equiv \exists y_1 \forall y_2 \psi(\bar{x}, y_1, y_2)$ and let $\bar{a} \in N$. We have $a_j < b_{i_0}$ for some i_0. Notice that $\mathcal{N} \models \varphi(\bar{a})$ if and only if

"there is some $b_{i_1} > b_{i_0}$ so that for all $b_{i_2} > b_{i_1}$ we have $\mathcal{N} \models \exists y_1 \forall y_2 (y_1 < b_{i_1}) \wedge (y_2 < b_{i_2}) \wedge \psi(\bar{a}, y_1, y_2)$ "

By HW this happens if and only if:

"there is some $b_{i_1} > b_{i_0}$ so that for all $b_{i_2} > b_{i_1}$ we have $\mathcal{M} \models \exists y_1 \forall y_2 (y_1 < b_{i_1}) \wedge (y_2 < b_{i_2}) \wedge \psi(\bar{a}, y_1, y_2)$ "

and since the b_i's are diagonal indiscernibles this happens if and only if

"$\mathcal{M} \models \exists y_1 \forall y_2 (y_1 < b_{i_0+1}) \wedge (y_2 < b_{i_0+2}) \wedge \psi(\bar{a}, y_1, y_2)$ "

We leave it as an exercise now to show that all induction axioms hold in $\mathcal{N}$. □

6.3. **Conclusion.**

Theorem 62. *The Paris-Harrington principle is not provable in Peano Arithmetic.*

Proof. By the previous subsection it suffice to show that $(\star)$ is not provable. Assume it was. Let $\mathcal{M}$ be any non-standard model of PA and fix c any infinite element, i.e., any $c \in M \setminus \mathbb{N}$.

Going now back to the combinatorics of the proof of Lemma 60: set $c = n = l$ and $m = 2c + 1$ so that we have $m > 2n$. Since the finite Ramsey theorem is provable in PA and since we assumed the same for $(\star)$ we can find a **least** w in M

so that $w \to (3c+1)^{2c+1}_{c+1}$ and then find the **least** s in M so that if $f_1, \ldots, f_c$ are c-many regressive maps from $[s]^{2c+1} \to s$, there is a $Y \subseteq (c, s)$ of size $|w|$ that is min-homogeneous for each f_j. The point here is that the formula which says F is the the Gödel code for k-many regressive functions can be used for $k = c$.

Following now the proof of Lemma 60 we obtain an infinite set $I \subseteq (c, s)$ with $|I| \geq c$ of diagonal indiscernibles for the first c-many Δ_0 formulas with variables $x_1, \ldots, x_c$ and $y_1, \ldots, y_c$ in any fixed Gödel coding.

Let $b_0 < b_1 < \ldots < b_n < \ldots$ be an initial segment of I. Since every actual Δ_0 formula is coded by some natural number in the standard part which is $< c$ we have that (b_n) is a diagonal sequence of indiscernibles for all Δ_0-formulas. Let $N = \{a \in M \mid \exists n\ a < b_n\}$. By previous lemma $\mathcal{N} := \mathcal{M}{\restriction}N$ is a model of PA. To finish the proof, it suffices to show that $\mathcal{N}$ does not satisfy $(\star)$. Notice that $c \in N$ while $s \notin N$.

Claim. The element w is in N.

Since finite Ramsey is provable in PA and $c \in N$ we can find some w' in N so that $w' \to (3c+1)^{2c+1}_{c+1}$ in $\mathcal{N}$. One checks now that because of the simplicity of the statement "p is a code for all colorings..." every $f\colon [w']^{2c+1} \to c+1$ that is coded by an element of M is actually coded by an element of N, same for subsets of w'. Therefore $w' \to (3c+1)^{2c+1}_{c+1}$ in $\mathcal{M}$ and by minimality of w we have that $w \leq w'$, i.e., $w \in N$.

Since $(\star)$ is assumed to be provable in PA then we should be able to find s' in N that in $\mathcal{N}$ performs the same function as s in $\mathcal{M}$. By a similar argument and the minimality of s we would get $s' \geq s$ contradicting that all b_n and therefore every element of N is below s. □

Mathematics Department, Caltech, 1200 E. California Blvd, Pasadena, CA 91125
Email address: panagio@caltech.edu
URL: http://www.its.caltech.edu/~panagio/