

ALTERNATING TURING MACHINES FOR INDUCTIVE LANGUAGES

DANIEL LEIVANT

Indiana University
e-mail address: leivant@indiana.edu

ABSTRACT. We show that alternating Turing machines, with a novel and natural definition of acceptance, accept precisely the inductive (Π_1^1) languages. Total alternating machines, that either accept or reject each input, accept precisely the hyper-elementary (Δ_1^1) languages. Moreover, bounding the permissible number of alternations yields a characterization of the levels of the arithmetical hierarchy. Notably, these results use simple finite computing devices, with finitary and discrete operational semantics, and neither the results nor their proofs make any use of transfinite ordinals.

Our characterizations elucidate the analogy between the polynomial-time hierarchy and the arithmetical hierarchy, as well as between their respective limits, namely polynomial-space and Π_1^1 .

1. INTRODUCTION

Inductive definitions via first-order positive operators constitute a broad computation paradigm. A fundamental result of computation theory, formulated in various guises over the last century, identifies the languages obtained by such definitions with those explicitly definable by Π_1^1 formulas, that is where second order quantification, over functions or relations, is restricted to positive occurrences of \forall . This central link was first discovered by Suslin in 1916 for sets of real numbers [16]. Kleene independently rediscovered the correspondence for sets of natural numbers (and so for languages) [6, 5]. Spector formulated the basic notions more explicitly [15], and Moschovakis, Barwise, and Gandy established the characterization for near-arbitrary countable first-order structures in 1971. This characterization of Π_1^1 in terms of inductive definability endows it with many of the structural properties of the computably enumerable (RE) sets, and suggests an analogy between computability based on finite processes, captured by Σ_1^0 , and a generalized form of computability based on infinite processes.

Our aim here is to capture the full power of inductive definability by a novel and natural definition of acceptance for alternating Turing machines. This is unrelated to notions of “infinite-time computations” that have been investigated repeatedly over the decades.

Alternation in computational and definitional processes is an idea that has appeared and reappeared in various guises over the last 50 odd years. Kleene’s definition of the arithmetical hierarchy

2012 ACM CCS: [Theory of computation]: Turing machines; Complexity classes; Higher order logic.

2010 Mathematics Subject Classification: 03D10, 03D60, 03D70.

Key words and phrases: Alternating Turing machines, inductive and hyper-elementary languages, arithmetical hierarchy, polynomial-time hierarchy.

This research was also supported by LORIA Nancy and Université Paris-Diderot.

in terms of quantifier alternation was an early manifestation, extended by Kleene, Spector, Gandy and others to the transfinite hyper-arithmetical hierarchy [14, 1]. An explicit link with alternation was discovered by Moschovakis [13, 12, 7], who characterized the inductive sets by a game quantifier [12, Theorem 5C2]. Harel and Kozen [3] showed how this characterization can be expressed in terms of an idealized programming language with random existential and universal assignments.

Alternating control made an entry into Computation Theory with the definition, by Chandra, Kozen and Stockmeyer, of alternating Turing machines [2], where existential and universal variants of non-determinism mesh. A state declared existential accepts when some child-configuration accepts, whereas a universal state accepts if all child-configurations accept. A computation can thus alternate between existential and universal phases. The striking result of [2], which has become a classic and has made its way into numerous textbooks, is that alternating Turing machines elucidate a powerful and elegant interplay between time and space complexity. Namely, for reasonable functions f the languages accepted by alternating Turing machines in time $O(f)$ are precisely the languages accepted by deterministic machines in space $O(f)$, and the languages accepted by alternating machines in space $O(f)$ are those accepted by deterministic machines in time $2^{O(f)}$. In particular, alternating polynomial time is precisely polynomial space. Moreover, when only fewer than k alternations are allowed, one obtains the k 'th level of the polynomial time hierarchy.

We establish here a formal parallel between the logical and the complexity-theoretic developments of alternation. Our point of departure is a simple and natural modification of the definition of acceptance by an alternating Turing machine, where acceptance by a universal configuration c will now refer to all configurations that end the universal computation-phase spawned by c , rather than just to the immediate children of c . We prove that a language is accepted by such a machine iff it is inductive (Π_1^1). Moreover, when up to k alternations are allowed, we obtain the k 'th levels of the arithmetical hierarchy. Also, if a language L is accepted by a machine which is total, in the sense that every input is either accepted or rejected, then L is hyper-arithmetical (Δ_1^1).

Note that our machines are no different from traditional alternating Turing machines: the difference lies only in the definition of acceptance. In particular, no infinitary rules, such as game quantifiers or random assignments, are used. We thus obtain here a direct correspondence between Π_1^1 and polynomial space, and between the arithmetical hierarchy and the polynomial-time hierarchy. The two sides of this correspondence are characterized by the same alternating Turing machines, but with a global (potentially infinitary) definition of acceptance for the former, and a local one for the latter.

The author is grateful to Yiannis Moschovakis for important comments on an early draft of this paper.

2. GLOBAL SEMANTICS FOR ALTERNATING COMPUTATIONS

2.1. Alternating Turing machines. The following will be used as reserved symbols, which we posit to occur only when explicitly referred to: \sqcup for the blank symbol, $+$ for the cursor-forward command, and $-$ for cursor-backward. We consider primarily single-tape machines. Given a finite alphabet Σ , an *alternating Turing machine (ATM) over Σ* is a device M consisting of

- (1) Disjoint finite sets E (existential states) and U (universal states). Elements of $Q = E \cup U$ are the *states*.
- (2) An element $s_0 \in Q$, dubbed the *start state*.
- (3) A finite alphabet $\Gamma \supseteq \Sigma \cup \{\sqcup\}$ (the *machine alphabet*).

(4) A relation $\delta \subseteq (Q \times \Gamma) \times (A \times Q)$, where $A = \Gamma \cup \{-, +\}$ is the set of *actions*.¹ δ may be construed as a multi-valued function, with domain $Q \times \Gamma$ and co-domain $A \times Q$. We write

$$q \xrightarrow[M]{\gamma(a)} q' \quad \text{for} \quad (q, \gamma, a, q') \in \delta, \text{ and omit the subscript } M \text{ when in no danger of confusion.}$$

A *configuration (cfg)* (of M) is a tuple (q, u, γ, v) with $q \in Q$, $u, v \in \Gamma^*$, and $\gamma \in \Gamma$. A cfg is said to be *existential* or *universal* according to the state therein. The definition of a yield relation $c \Rightarrow c'$ between configurations is defined as usual; that is, it is generated inductively by the conditions:²

- If $q \xrightarrow[M]{\gamma(+)} q'$ then $(q, u, \gamma, \tau v) \Rightarrow (q', u\gamma, \tau, v)$ and $(q, u, \gamma, \varepsilon) \Rightarrow (q', u\gamma, \sqcup, \varepsilon)$;³
- If $q \xrightarrow[M]{\gamma(-)} q'$ then $(q, u\tau, \gamma, v) \Rightarrow (q', u, \tau, \gamma v)$ and $(q, \varepsilon, \gamma, v) \Rightarrow (q', \varepsilon, \gamma, v)$ (i.e. the cursor does not move); and
- If $q \xrightarrow[M]{\gamma(\tau)} q'$ then $(q, u, \gamma, v) \Rightarrow (q', u, \tau, v)$.

Following [8] we dispense here with accepting and rejecting states: when no transition applies to a universal cfg then it has no children, and so the condition for acceptance is satisfied vacuously. Dually, a dead-end existential cfg is rejecting. For brevity we also write configurations (q, u, γ, v) as a pairs (q, w) , where the understanding is that w is a “cursored string” $u\underline{\gamma}v$.

2.2. Acceptance and rejection. The *computation tree of M for cfg c* is a finitely-branching (but potentially infinite) tree $T_M(c)$ of cfg-occurrences $\langle \alpha, c \rangle$, α being the node-address and c the cfg, where the children of $\langle \alpha, c \rangle$ are $\langle i\alpha, c_i \rangle$ with c_i the i -th cfg c' such that $c \Rightarrow c'$ (under some fixed ordering of the transition rules of δ).

We write $c \xrightarrow[\exists]{\rightarrow} c'$ when $c \Rightarrow c'$ and c is existential, $c \xrightarrow[\exists]{\twoheadrightarrow} c'$ if $c \xrightarrow[\exists]{\rightarrow^*} c'$ and c' is universal. (As usual, $\xrightarrow[\exists]{\rightarrow^*}$ is the reflexive and transitive closure of $\xrightarrow[\exists]{\rightarrow}$.) In other words, the universal cfg c' can be reached from the cfg c by successive applications of the yield relation \Rightarrow , where all intermediate states are existential.

The definitions of $c \xrightarrow[\forall]{\rightarrow} c'$ and $c \xrightarrow[\forall]{\twoheadrightarrow} c'$ are similar. We call a cfg c' as above, for either $\xrightarrow[\exists]{\twoheadrightarrow}$ or $\xrightarrow[\forall]{\twoheadrightarrow}$, an *alternation-pivot* (for c).

The set AC of *accepted configurations* is generated inductively by the following closure conditions:

- (1) If c is existential and $c' \in AC$ for some c' such that $c \xrightarrow[\exists]{\twoheadrightarrow} c'$, then $c \in AC$.
- (2) If c is universal and $c' \in AC$ for all c' such that $c \xrightarrow[\forall]{\twoheadrightarrow} c'$, then $c \in AC$.

If S is any set of configurations, we write $CC[S]$ for the conjunction of the conditions above for S . That is,

- (1) If c is existential and $c' \in S$ for some c' such that $c \xrightarrow[\exists]{\twoheadrightarrow} c'$, then $c \in S$.
- (2) If c is universal and $c' \in S$ for all c' such that $c \xrightarrow[\forall]{\twoheadrightarrow} c'$, then $c \in S$.

Thus, AC is generated by the closure conditions $CC[AC]$. Note that $CC[S]$ is a Π_2^0 formula. For instance, (2) can be expressed as

$$\forall \text{ cfg } c \quad ((\forall \text{ traces witnessing a relation } c \xrightarrow[\forall]{\twoheadrightarrow} c') \quad c' \in S) \quad \rightarrow c \in S$$

¹We follow here the convention whereby Turing machines either move their cursor or overwrite it, but not both.

²Note that inductive definitions posit implicitly an exclusivity condition, so the “only if” direction is not needed.

³We write ε for the empty string.

Thus, the set AC of accepted configurations is explicitly definable as the set of configurations c satisfying the Π_1^1 formula

$$\forall S (CC[S] \rightarrow c \in S)$$

Similarly, the set RC of *rejected configurations* is generated inductively by closure conditions dual to the ones above:

- (1) If c is existential and $c' \in RC$ for all c' such that $c \xrightarrow{\exists} c'$, then $c \in RC$.
- (2) If c is universal and $c' \in RC$ for some c' such that $c \xrightarrow{\forall} c'$, then $c \in RC$.

Again, RC is explicitly definable by a Π_1^1 formula.

We say that a state is *dead-end* if no transition rule applies to it. A universal dead-end state is an *accept-state*, and an existential dead-end state a *reject-state*.

The *initial configuration* of the machine M for input w is $\langle s_0, \varepsilon, \sqcup, w \rangle$. M *accepts an input string* w if the initial cfg for w is in the set AC of accepted configurations, as defined above. Dually, M *rejects* w if that cfg is in RC . For example, if M has only universal states, then no computation tree can have an alternation-pivot, and so every w is accepted. The computation tree for w may well have leaves, that is dead-end configurations, but since here these are all universal configurations with no children, they are accepted. Dually, if M has only existential states, then no input can be accepted. These examples are merely consequences of our choice to represent acceptance and rejection by dead-end universal and existential configurations, respectively. For example, a usual non-deterministic Turing machine can be obtained simply by considering each accept-state as a universal state with no applicable transition rule.

The *language accepted by an ATM* M is

$$\mathcal{L}(M) = \{w \in \Sigma^* \mid M \text{ accepts } w\}$$

and the *language rejected by* M is

$$\bar{\mathcal{L}}(M) = \{w \in \Sigma^* \mid M \text{ rejects } w\}$$

It is easy to see that $\mathcal{L}(M) \cap \bar{\mathcal{L}}(M) = \emptyset$. Our definitions of acceptance and rejection of configurations conform to the local closure conditions of acceptance (and rejection) of usual ATMs, as we point out in the next Proposition. However, those conditions cannot be used to *define* acceptance and rejection, because we allow infinite computation trees.

Proposition 2.1. *Let M be an ATM, T a computation tree of M for input w . If c is a cfg in the tree, with children $c_1 \dots c_m$, then*

- (1) *If c is existential, then c is accepted iff some c_i is accepted, and c is rejected iff all c_i 's are rejected.*
- (2) *If c is universal, then c is accepted iff all c_i 's are accepted, and c is rejected iff some c_i is rejected.*

Proof. Let c be existential. If c is an accepted cfg, i.e. $c \xrightarrow{\exists} c'$ for some accept-state c' , then $c_i \xrightarrow{\exists}^* c'$ for some c_i , since c itself is existential. If that c_i is existential, then it is accepted, by definition; and if it is not, then $c_i = c'$, which is accepted by assumption.

Conversely, suppose that some c_i is accepted. If c_i is universal, then $c \xrightarrow{\exists} c'$, and so c is accepted, by definition of acceptance. If c_i is existential, then there must be an accepted c' such that $c_i \xrightarrow{\exists}^* c'$; but then $c \xrightarrow{\exists}^* c'$, so c is accepted.

Other cases are proved similarly. □

2.3. Divergence and totality. An ATM may well neither accept nor reject an input string w . For example, if the computation tree of M for a given input w has infinitely many alternation-pivots along each computation-trace (a situation that we can engineer fairly easily), then M neither accepts nor rejects that input. Indeed, the empty set satisfies the closure conditions for acceptance of w , as well as the closure conditions for rejection.

We say that an ATM M is *total* if every input is either accepted or rejected by M . Let us identify a simple condition that guarantees totality. We say that a computation tree is *alternation well-founded* if no branch has infinitely many alternation-pivots. An ATM is *alternation well-founded* if all its computations are alternation well-founded.

Proposition 2.2. *If an ATM is alternation well-founded then it is total.*

Proof. We prove the contra-positive: if a cfg c is neither accepted nor rejected, then the computation tree T that it spawns has a branch with infinitely many alternation-pivots.

Suppose c is universal. Since c is not accepted, we must have $c \xrightarrow{\forall} c'$ for some alternation-pivot c' which is not accepted. And since c is also not rejected, all of its alternation-pivots, and in particular c' , are not rejected. If c is existential, a dual argument shows that $c \xrightarrow{\forall} c'$ for some alternation-pivot c' which is neither accepted nor rejected.

Iterating the argument we obtain a branch with an infinite sequence $c_0 = c, c_1 = c', \dots$ of successive alternation-pivots, all of which are neither accepted nor rejected. \square

The converse of Proposition 2.2 fails. Indeed, it is easy to construct a total ATM that is not alternation well-founded, by inserting innocuous computation traces with infinitely many alternation-pivots, with no impact on the acceptance or rejection of the input. See the proof of Proposition 2.4 below.

2.4. Duality and one-sidedness. The *dual* of an ATM M is the machine \bar{M} whose transition relation is that of M , but with the sets of universal and existential states interchanged, that is with M 's sets U and E as the sets of existential and universal states, respectively.

Directly from the definitions we have

Proposition 2.3. *Let \bar{M} be the dual of M . Then $\mathcal{L}(\bar{M}) = \bar{\mathcal{L}}(M)$, whence also $\bar{\mathcal{L}}(\bar{M}) = \mathcal{L}(M)$. \square*

A machine M is *one-sided* if it either has no accepted configurations, or no rejected configurations.

Proposition 2.4. *For every machine M there are one-sided machines M^+ and M^- such that $\mathcal{L}(M) = \mathcal{L}(M^+)$, and $\bar{\mathcal{L}}(M) = \bar{\mathcal{L}}(M^-)$.*

Proof. The proof is analogous to the conversion of a deterministic TM to a TM that diverges for any input it does not accept.

Let M^+ be obtained from M by expanding its transition relation as follows. Using auxiliary states and transitions, we add for every existential state a transition into an auxiliary universal state that starts an infinite trace (using auxiliary states) of alternation-pivots. That is, we create a fresh alternation-pivot following each existential cfg, where that alternation-pivot is neither accepted nor rejected. Each state accepted in M is accepted in M^+ , because no existential configuration is losing any pivot by the modification. And if a state is accepted in M^+ , then it is accepted in M , because the set A of configurations of M^+ that consists just of the accepting configurations of M satisfies the closure conditions AC for M^+ , and therefore contains the set of configurations accepted by M^+ (which is the minimal such set).

But M^+ has no rejected configurations: existential configurations cannot be rejected because they have an alternation-pivot, namely the one introduced by the definition of M^+ , which is not rejected. And then universal configurations cannot be rejected, because all their alternation-pivots, which are existential, are non-rejected.

The construction of M^- is dual. □

2.5. The Arithmetical Hierarchy. We say that an ATM M is Σ_k if its initial state is existential, and for every $w \in \Sigma^*$, all branches of the computation-tree for w have $\leq k$ alternation-pivots. The definition of Π_k machines is similar, but with a universal initial state. Here again we posit that the existential states of Π_1 machines have no applicable transition rules.

Theorem 2.5. *Let $k \geq 1$. A language is Σ_k^0 (Π_k^0) iff it is accepted by a Σ_k (Π_k , respectively) ATM.*

Proof. The proof is by induction on k . For the base case Σ_1^0 , let L be a language defined by a Σ_1^0 formula, that is

$$L = \{x \in \Sigma^* \mid \varphi[x]\}$$

where

$$\varphi[x] \equiv \exists w_1, \dots, w_r \varphi_0[\vec{w}, x]$$

with φ_0 a bounded formula, i.e. with all quantifiers bounded (under the substring relation). Define a Σ_1 machine M that accepts L , as follows. M branches existentially to choose a string $w = w_1\# \dots \#w_r$, then proceeds to check deterministically that $\varphi_0[w_1 \dots w_r]$. (We classify the states for that deterministic process to be universal, so that dead-end states are accepted.)

For the converse, note first that in a Σ_1 computation tree the universal configuration are all accepted, since they have no pivots. So acceptance by a Σ_1 machine M is definable by the Σ_1^0 formula that states, for input w , the existence of a finite tree of configurations, related by the rules of M , with the initial configuration for w as root, of which the internal nodes are existential and the leaves are universal.

For the base case Π_1^0 , suppose L is defined by a Π_1^0 formula

$$\varphi[x] \equiv \forall w_1 \dots w_r \varphi_0[w_1, \dots, w_r, x]$$

Define a Π_1 machine M that accepts L , as follows. M generates strings $w_1\# \dots \#w_r$ in successive lexicographic order. After each such choice M branches universally to the next string as well as to a deterministic module that accepts x iff $\varphi_0[\vec{w}, x]$ for the current value of $w_1\# \dots \#w_r$.

Conversely, if $L = \mathcal{L}(M)$ where M is an Π_1 machine, then L is definable by a formula that states that for all (finite) computation traces, the trace's last configuration is not existential (i.e. rejected).

The induction step generalizes the induction basis: The properties above are proved for level $k+1$ of the Arithmetical Hierarchy by referring to sub-computations at level k , rather than to deterministic sub-computations. □

3. ALTERNATION AND INDUCTIVE LANGUAGES

3.1. Accepted languages are inductive. Fix an alphabet Σ . Consider formulas over the vocabulary (i.e. similarity type) with an identifier for each letter in Σ as well as for the empty-string, a binary function-identifier for concatenation, and a binary relation for the substring relation.

Proposition 3.1. *The following conditions are equivalent for a language $L \subseteq \Sigma^*$.*

I1: L is defined by a formula of the form $\forall f \varphi[w, f]$, where φ is first-order and f ranges over $\Sigma^* \rightarrow \Sigma^*$.

I2: L is defined by a formula of the form $\forall f \exists x \varphi_0[w, f, x]$, where φ_0 is a bounded formula, i.e. with each quantifier restricted to substrings of some string.

I3: L is defined by a formula of the form $\forall f \exists x \varphi_0[w, \bar{f}(x), x]$, where $\bar{f}(x)$ abbreviates the string $f(0)\# \cdots \# f(|x|)$ (with $\#$ a fresh symbol, used as a textual separator).

I4: L is defined by a formula of the form $\forall S \exists x \forall y \varphi_0[w, f, x, y]$, where S ranges over subsets of Σ^* .

Proof. I1 implies I2 by the Kuratowski-Tarski algorithm [11]. I2 implies I3 by the boundedness of φ_0 . I1 implies I4 by an interpretation of functions by relations (and hence sets, since we are talking about languages), and I3 and I4 each implies I1 trivially. \square

Note that the use of a set quantifier in I4 necessitates an alternation of first-order quantifiers, which is not needed in I1. This is essential: without the presence of the first-order universal quantifier $\forall y$ we get Kreisel's *strict- Π_1^1* formulas, which are no more expressive than Σ_1^0 [9, 10].

A language $L \subseteq \Sigma^*$ is *inductive* (Π_1^1) when it satisfies the equivalent conditions of Proposition 3.1 (see e.g. [4]).

Recall that our definition above of acceptance by an ATM refers to the set AC of accepted configurations, which is Π_1^1 definable. We therefore have:

Proposition 3.2. *Every language accepted by an ATM is inductive.*

3.2. Inductive languages are accepted.

Proposition 3.3. *Every inductive language is accepted by an ATM.*

Proof. We refer to characterization (I3) of Π_1^1 languages. As usual, Σ^n stands for the set of strings over Σ of length n . Let L be a language defined by

$$\forall f \exists x \varphi_0[w, \bar{f}(x), x]$$

which we write momentarily as

$$\forall f \exists x \varphi_0[w, z_0 \# \cdots \# z_n, x]$$

where $n = |x|$ and $z_i = f(i)$. This is equivalent to the following infinite formula (where, as usual, Σ^n is the set of strings of length n).

$$\begin{aligned} & \varphi_0[w, \varepsilon, \varepsilon] \\ & \vee \forall z_0 (\exists x \in \Sigma^1 \varphi_0[w, z_0, x]) \\ & \vee \forall z_1 (\exists x \in \Sigma^2 \varphi_0[w, z_0 \# z_1, x]) \\ & \vee \forall z_2 (\exists x \in \Sigma^3 \varphi_0[w, z_0 \# z_1 \# z_2, x]) \\ & \vee \forall z_3 (\exists x \in \Sigma^4 \varphi_0[w, z_0 \# z_1 \# z_2 \# z_3, x]) \\ & \vee \dots \end{aligned} \tag{3.1}$$

We use here infinitary formulas for informal expository purpose; compare [12, 13].

Formula (3.1) is captured by an ATM which, on input w ,

- (1) checks deterministically $\varphi_0[w, \varepsilon, \varepsilon]$; if this fails,
- (2) chooses by universal nondeterminism a value z_0 ;⁴

⁴Recall from the introduction that such a choice, for our finitely-branching machine, involves a computation tree with an infinite branch.

- (3) for each such choice for z_0 , branches by existential nondeterminism to
- (a) guess (by existential nondeterminism) an $x \in \Sigma$, then check (deterministically) $\varphi_0[w, z_0, x]$; if this fails
 - (b) choose by universal nondeterminism a z_1 ;
 - (c) etc. □

Combining Propositions 3.2 and 3.3 we conclude:

Theorem 3.4. *L is inductive iff it is accepted by an ATM.*

4. TOTAL MACHINES AND HYPER-ARITHMETICAL LANGUAGES

A basic result of computation theory is the characterization of decidable languages in terms of semi-decidability:

Theorem 4.1. *A language $L \subseteq \Sigma^*$ is accepted by a Turing machine that terminates for all input iff both L and its complement are accepted by a Turing machine.*

The analog of Theorem 4.1 is

Theorem 4.2. *A language $L \subseteq \Sigma^*$ is accepted by a total ATM iff both L and its complement $\bar{L} = \Sigma^* - L$ are accepted by an ATM.*

The forward implication of the Theorem is easy: If a language L is accepted by a total ATM M then the dual machine \bar{M} accepts \bar{L} , by Proposition 2.3.

Towards proving below the converse implication, assume that $L = \mathcal{L}(M_0)$ and $\bar{L} = \mathcal{L}(N)$. By Proposition 2.4 we may assume that neither machine has rejected configurations. Thus \bar{L} is rejected by the machine $M_1 = \bar{N}$, which has no accepted cfg. We wish to construct out of M_0 and M_1 a total machine M that accepts L and rejects \bar{L} . A naive emulation of the standard proof of Theorem 4.1 would swap control between M_0 and M_1 after each computation step. That is, M is defined as a two-tape machine, whose states are tuples $\langle q_0, q_1, j \rangle$, with q_i a state of M_i , and where $j \in \{0, 1\}$ indicates which machine is to make a move. The type of $\langle q_0, q_1, j \rangle$ (existential or universal) is the type of q_j . Since M_1 has no accepted cfg, a cfg c of M would be accepted when its M_0 component is accepted by M_0 ; and since M_0 has no rejected configurations, c would be rejected in M if its M_1 component is rejected by M_1 .

However, the construction above does not work for our ATMs, due to the global definition of acceptance. Consider a universal cfg c_0 of M_0 , which is accepted in M_0 because it has no pivots. When c_0 is coupled in M with a universal cfg c_1 of M_1 , the combined cfg may spawn a computation tree with pivots of M_1 , whose M_0 -component is not accepted in M_0 . The combined cfg is not accepted then in M , even though c_0 is accepted in M_0 .

We consider instead a merge of M_0 and M_1 where control swap from a universal phase of M_0 to M_1 is delayed until that phase has ended, and dually for an existential phase of M_1 .

Note that for simple Turing machines (deterministic or nondeterministic) phases coincide with computation steps, since no universal configurations are present.

More precisely, we posit, without loss of generality, that M_0 and M_1 are single-tape ATM's over a common input alphabet Σ , and using a common extended alphabet $\Gamma \subset \Sigma \cup \{\sqcup\}$. The combined machine M is then defined as follows.

- M is a two-tape ATM, whose states of interest are tuples $\langle q_0, q_1, j \rangle$, with q_i a state of M_i , ($i = 0, 1$). The type of $\langle q_0, q_1, j \rangle$ (existential or universal) is the type of q_j (in M_j).

- In addition, M has auxiliary states and (deterministic) transitions that pre-process its computation by copying the input into the second tape, reinitializing the cursor positions, and passing control to a state $\langle s_0, s_1, 0 \rangle$, where s_i is the initial state of M_i .
- For $\gamma, \delta \in \Gamma$, $\alpha \in \{+, -\} \cup \Gamma$,
if $q_0 \xrightarrow{\gamma(\alpha)} p_0$ is a rule of M_0 then
 - If both q_0 and p_0 are universal, then

$$\langle q_0, q_1, 0 \rangle \xrightarrow{\gamma, \delta(\alpha, \delta)} \langle p_0, q_1, 0 \rangle$$

i.e. on reading γ on the first tape, and δ on the second, M performs action α on component 0 of the cfg, action δ (i.e. no-op) on component 1, and leaves control to component 0.

- Otherwise, i.e. if at least one of q_0, p_0 is existential, then

$$\langle q_0, q_1, 0 \rangle \xrightarrow{\gamma, \delta(\alpha, \delta)} \langle p_0, q_1, 1 \rangle$$

- If $q_1 \xrightarrow{\gamma(\alpha)} p_1$ is a rule of M_1 , then
 - If both q_1 and p_1 are existential, then

$$\langle q_0, q_1, 1 \rangle \xrightarrow{\gamma, \delta(\alpha, \delta)} \langle q_0, p_1, 1 \rangle$$

- Otherwise, i.e. if at least one of q_1, p_1 is universal, then

$$\langle q_0, q_1, 1 \rangle \xrightarrow{\gamma, \delta(\alpha, \delta)} \langle q_0, p_1, 0 \rangle$$

Proposition 4.3. *Assume that no string is both accepted by M_0 and rejected by M_1 . Then M accepts every string accepted by M_0 .*

Proof. We prove that if M_0 accepts a cfg (q_0, u_0) then, for every non-rejected cfg (q_1, u_1) of M_1 , M accepts $(\langle q_0, q_1, 0 \rangle, \langle u_0, u_1 \rangle)$. If M_0 accepts u , then (by assumption) M_1 does not reject it, and so the Proposition follows by considering the cfg $(\langle s_0, s_1, 0 \rangle, \langle u, u \rangle)$.

Define the set A of M_0 -configurations by

$$A = \{ \langle q_0, u_0 \rangle \mid (\langle q_0, q_1, 0 \rangle, \langle u_0, u_1 \rangle) \text{ is accepted in } M \\ \text{for all non-rejected configurations } (q_1, u_1) \text{ of } M_1 \}$$

We show that A satisfies the closure conditions defining the set of configurations accepted by M_0 .

- **The existential closure condition:** Suppose that $(q_0, u_0) \xrightarrow{\exists} (p_0, w_0)$, where $(p_0, w_0) \in A$, and the reduction sequence is of length $n \geq 1$.⁵ We prove that $(q_0, u_0) \in A$ by induction on n . Let $(q_0, u_0) \xrightarrow{\exists} (r_0, v_0) \xrightarrow{\exists} (p_0, w_0)$. Note, first, that we must have $(r_0, v_0) \in A$: if $n = 1$ then $(r_0, v_0) = (p_0, w_0) \in A$, and if $n > 1$ then $(r_0, v_0) \in A$ by IH.

Towards proving that $(q_0, u_0) \in A$ let (q_1, u_1) be a non-rejected cfg of M_1 . We have

$$(\langle q_0, q_1, 0 \rangle, \langle u_0, u_1 \rangle) \xrightarrow{\exists} (\langle r_0, q_1, 1 \rangle, \langle v_0, u_1 \rangle)$$

We show that $(\langle r_0, q_1, 1 \rangle, \langle v_0, u_1 \rangle)$ is accepted in M , whence so is $(\langle q_0, q_1, 0 \rangle, \langle u_0, u_1 \rangle)$.

We have the following cases.

- q_1 is universal. Each M_1 -cfg (r_1, v_1) such that $(q_1, u_1) \xrightarrow{\forall} (r_1, v_1)$ must be non-rejected, since (q_1, u_1) is non-rejected. We have

$$(\langle r_0, q_1, 1 \rangle, \langle v_0, u_1 \rangle) \xrightarrow{\forall} (\langle r_0, r_1, 0 \rangle, \langle v_0, v_1 \rangle)$$

⁵ $n = 0$ is excluded, since by definition of $\xrightarrow{\exists}$ the state q_0 is existential and p_0 is universal.

and the latter cfg is accepted, since $(r_0, v_0) \in A$ and (r_1, v_1) is non-rejected. It follows that $(\langle r_0, q_1, 1 \rangle, \langle v_0, u_1 \rangle)$ is accepted in M .

- q_1 is existential. Since (q_1, u_1) is non-rejected, it follows that $(q_1, u_1) \xrightarrow{\exists} (r_1, v_1)$ for some non-rejected cfg (r_1, v_1) of M_1 . By definition of M , we have

$$(\langle r_0, q_1, 1 \rangle, \langle v_0, u_1 \rangle) \xrightarrow{\exists} (\langle r_0, r_1, 0 \rangle, \langle v_0, v_1 \rangle)$$

The latter cfg is accepted, since $(r_0, v_0) \in A$, and (r_1, v_1) is non-rejected. It follows that $(\langle r_0, q_1, 1 \rangle, \langle v_0, u_1 \rangle)$ is accepted in M .

We have thus shown that if $(q_0, u_0) \xrightarrow{\exists} (p_0, w_0)$, where $(p_0, w_0) \in A$, then $(q_0, u_0) \in A$.

- **The universal closure condition:** Suppose that for all (p_0, w_0) , if $(q_0, u_0) \xrightarrow{\forall} (p_0, w_0)$ then $(p_0, w_0) \in A$. Towards showing that $(q_0, u_0) \in A$, let (q_1, u_1) be a non-rejected cfg of M_1 .

By definition of M , if $(\langle q_0, q_1, 0 \rangle, \langle u_0, u_1 \rangle) \xrightarrow{\forall} C$, where C is a cfg of M , then $C = (\langle p_0, q_1, 1 \rangle, \langle w_0, u_1 \rangle)$, where $(q_0, u_0) \xrightarrow{\forall} (p_0, w_0)$.

We show that $(\langle p_0, q_1, 1 \rangle, \langle w_0, u_1 \rangle)$ is accepted in M for each such (p_0, w_0) , implying that $(\langle q_0, q_1, 0 \rangle, \langle u_0, u_1 \rangle)$ is accepted.

We have the following cases.

- q_1 is universal. Suppose $(q_1, u_1) \xrightarrow{\forall} (p_1, v_1)$. Then

$$(\langle p_0, q_1, 1 \rangle, \langle w_0, u_1 \rangle) \xrightarrow{\forall} (\langle p_0, p_1, 0 \rangle, \langle w_0, v_1 \rangle)$$

The cfg (p_1, v_1) must be non-rejected, since (q_1, u_1) is non-rejected. Since $(p_0, w_0) \in A$, it follows that $(\langle p_0, p_1, 0 \rangle, \langle w_0, v_1 \rangle)$ is accepted. This being the case for every (p_1, v_1) as above, we conclude that $(\langle p_0, q_1, 1 \rangle, \langle w_0, u_1 \rangle)$ is accepted.

- q_1 is existential. Since (q_1, u_1) is non-rejected, it follows that $(q_1, u_1) \xrightarrow{\exists} (p_1, v_1)$ for some non-rejected configuration (p_1, v_1) of M_1 . By definition of M , we have

$$(\langle p_0, q_1, 1 \rangle, \langle w_0, u_1 \rangle) \xrightarrow{\exists} (\langle p_0, p_1, 0 \rangle, \langle w_0, v_1 \rangle)$$

The latter configuration is accepted in M , since $(p_0, w_0) \in A$ and (p_1, v_1) is non-rejected. It follows that $(\langle p_0, q_1, 1 \rangle, \langle w_0, u_1 \rangle)$ is also accepted.

We have thus shown that if $(p_0, w_0) \in A$ whenever $(q_0, u_0) \xrightarrow{\forall} (p_0, w_0)$, then $(q_0, u_0) \in A$, that is A satisfies the universal closure condition for acceptance in M_0 .

Since A satisfies both the existential and the universal closure conditions for acceptance in M_0 , it follows that A contains every accepting cfg of M_0 , proving the Proposition. \square

Proof of Theorem 4.2 — Concluded. We have noted already the forward implication. We show that if L and \bar{L} are accepted by ATM's, then L is accepted by a total ATM.

Let $L = \mathcal{L}(M_0)$ and $\bar{L} = \mathcal{L}(N)$, and refer to the machines M_1 and M of the discussion above. By Proposition 4.3, M accepts every string accepted by M_0 .

An argument dual to that in the proof of Proposition 4.3 shows that M rejects every cfg $(\langle q_0, q_1 \rangle, \langle u_0, u_1 \rangle)$, where M_1 rejects (q_1, u_1) and M_0 does not accept (q_0, u_0) . In particular, assuming M_1 rejects an input string u , M rejects $(\langle t_0, s_1, 1 \rangle, \langle v, u \rangle)$ whenever (t_0, v) is a non-accepted configuration of M_0 .

A small extra step is needed to account for the fact that M_0 , and not M_1 , has the initial control in M . Posit, without loss of generality, that the initial state s_0 of M_0 is existential and deterministic (i.e. at most one transition applies). Since M_0 does not accept u , we must have $(s_0, u) \xrightarrow{\exists} (t_0, v)$

where (t_0, v) is a non-accepted cfg. But then the unique initial transition of M (past the initialization phase) is

$$(\langle s_0, s_1, 0 \rangle, \langle u, u \rangle) \xrightarrow{\exists} (\langle t_0, s_1, 1 \rangle, \langle v, u \rangle)$$

Since M_1 rejects (s_1, u) and M_0 does not accept (t_0, v) , M must reject $(\langle t_0, s_1, 1 \rangle, \langle v, u \rangle)$, as noted above, and therefore must also reject $(\langle s_0, s_1, 0 \rangle, \langle u, u \rangle)$.

In conclusion, M accepts every string accepted by M_0 , and rejects every string rejected by M_1 . So M is a total machine that accepts L and rejects \bar{L} . \square

5. CONCLUSION

The combined use of existential and universal nondeterminism has been of interest primarily in Computational Complexity theory, but has not been considered thus far as a tool in the foundations of computing. This is because the semantics of acceptance has been defined “locally”, that is in terms of the relation between computational configurations and their immediate descendants. That semantics implies that acceptance (and rejection) are witnessed by finite computation trees, and thus cannot lead us beyond the semi-decidable (RE) languages. Viewed from another angle, the closure properties involved are Π_1^0 , and so the accepted languages are defined by strict- Π_1^1 formulas (see §3.1 above).

We showed here that a very natural alternative semantics for universal nondeterminism changes the picture radically, as the languages accepted are precisely the Π_1^1 ones. This further illustrates the foundational analogy between alternation in feasible time with local semantics, which yields PSpace as a limit of the PTime Hierarchy (starting with PTime and NP), and alternation for arbitrary computations with global semantics, which yields Π_1^1 as a limit of the arithmetical hierarchy (starting with Σ_1^0).

Generalized models of computation that go beyond computability have been studied extensively, of course. The novelty of the approach here is that it refers to the very same hardware as traditional Turing machines (albeit with both modes of nondeterminism), but redefines the notion of acceptance, in a way that remains consistent with the underlying, intuitive, intent.

The ability to refer to both computational complexity and higher recursion theory using the same machine models has the potential of suggesting analogies between results, and thereby transfer of results. We believe that this will provide insights and additional machine-based proofs for Higher Recursion Theory.

The approach developed here seems to also break with past works in this area in that it dispenses with transfinite recurrence and induction over Kleene’s constructive ordinals, and does not use any transfinite stage-comparison technique. Instead, the proofs use inductive definitions directly.

Directly dealing with inductive definitions, without calibrating them by ordinals provides, in fact, a closer analogy with finite computing. Computation traces of machines and of programs are construed intuitively as finite objects, without direct reference to the natural numbers, either as clocking computation steps or as codes for computational objects. With the frequent use of “structural induction” and “structural recurrence.” It is, therefore, natural to expect that higher-order computation traces can be studied directly, without a detour through transfinite clocking by constructive ordinals. The proof of Theorem 4.2 achieves precisely that.

REFERENCES

- [1] Jon Barwise. *Admissible Sets and Structures*, volume 7 of *Perspectives in Mathematical Logic*. Springer-Verlag, Berlin, 1975.
- [2] Ashok Chandra, Dexter Kozen, and Larry Stockmeyer. Alternation. *Journal of the ACM*, 28:114–133, 1981.
- [3] David Harel and Dexter Kozen. A programming language for the inductive sets, and applications. *Information and Control*, 63:118–139, 1984.
- [4] Stephen C. Kleene. *Introduction to Metamathematics*. Wolters-Noordhof, Groningen, 1952.
- [5] Stephen C. Kleene. Hierarchies of number theoretic predicates. *Bull. American Mathematical Society*, 61:193–213, 1955.
- [6] Stephen C. Kleene. On the form of predicates in the theory of constructive ordinals. *American journal of mathematics*, 77:405–428, 1955.
- [7] Phokion Kolaitis. Game quantification. In *Model-Theoretic Logics*, pages 365–421. Springer-Verlag, New York, 1985.
- [8] Dexter Kozen. *Theory of Computation*. Springer, London, 2006.
- [9] G. Kreisel. La prédictivité. *Bull. Soc. math. France*, 88:371–391, 1960.
- [10] Georg Kreisel. Survey of proof theory. *Journal of symbolic Logic*, 33:321–388, 1968.
- [11] Kazimierz Kuratowski and Alfred Tarski. Les opérations logiques et les ensembles projectifs. *Fund. Math.*, 17:240–248, 1931.
- [12] Y. Moschovakis. *Elementary Induction on Abstract Structures*. North-Holland, Amsterdam, 1974.
- [13] Yianis Moschovakis. The game quantifier. *Proc. AMS*, 31:245–250, 1971.
- [14] H. Rogers. *Theory of Recursive Functions and Effective Computability*. McGraw-Hill, New York, 1967.
- [15] Clifford Spector. Inductively defined sets of natural numbers. In *Infinite methods*, pages 97–102. Pergamon, New York, 1961.
- [16] Mikhail Yakovlevich Suslin. Sur une définition des ensembles mesurables B sans nombres transfinis. *Comptes rendus de l'Académie des sciences*, 164:88–91, 1917.