Alternating Turing Machines and the Analytical Hierarchy

Daniel Leivant*

Indiana University
leivant@indiana.edu

Abstract

We use notions originating in Computational Complexity to provide insight into the analogies between Computational Complexity and Higher Recursion Theory. We consider alternating Turing machines, but with a modified, global, definition of acceptance. We show that a language is accepted by such a machine iff it is inductive (Π^1_1). Moreover, total alternating machines, which either accept or reject each input, accept precisely the hyper-arithmetical (Δ^1_1) languages. Also, bounding the permissible number of alternations we obtain a characterization of the levels of the arithmetical hierarchy.

The novelty of these characterizations lies primarily in the use of finite computing devices, with finitary, discrete, computation steps. We thereby elucidate the analogy between the polynomial-time and the arithmetical hierarchies, as well as between their respective limits, namely the classes of the polynomial-space and Π^1_1 languages.

Keywords: Alternating Turing machines, inductive and hyper-arithmetical sets, arithmetical hierarchy, polynomial-time hierarchy, analytical hierarchy.

1 Introduction

Alternation in computational and definitional processes is an idea that has appeared and reappeared in many guises over the last 50 odd years. Kleene's definition of the arithmetical hierarchy in terms of quantifier alternation was an early manifestation, extended by Kleene, Spector, Gandy and others to the transfinite hyper-arithmetical hierarchy [12, 1].

An even more explicit link with alternation was discovered by Moschovakis [11, 10, 5], who characterized the inductive sets by a game quantifier [10, 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.

Alternation made an entry into Computational Complexity with the definition by Chandra, Kozen and Stockmeyer of alternating Turing machines [2], where disjunctive and conjunctive variants of non-determinism co-exist. 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 made its way to numerous textbooks, is that alternating Turing machines provide a powerful interplay between time and space complexity: 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 up to k alternations are allowed, one obtains the k'th level of the polynomial time hierarchy.

*This research was also supported by LORIA Nancy and PPS Université Paris Diderot

We establish here a direct link 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 refers to all configurations that end the universal computation-phase spawned by c, rather than to the immediate children of c only. We prove that a language is accepted by such a machine iff it is inductive (Π^1_1). Moreover, when only up to k alternations are allowed, we obtain the k'th level 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. As a result, the characterization above of Δ^1_1 has a far simpler proof than for previous characterizations, such as the programming language IND of [3]. Indeed, the latter is basically a recasting of inductive definability using random assignments to convey first-order quantification; in particular, the proof there of a result analogous to 10 below, uses inductive definability and stage-comparison.

We thus obtain here a simple and 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.

We hope that this work will lead to new insights into classical results of Higher Recursion Theory, as did decades ago the characterizations in terms of inductive definitions [1] and game quantifiers.

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. For brevity 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 $s_0 \in Q$, called 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] One can construe δ as a multi-valued function, with domain $Q \times \Gamma$ and co-domain $A \times Q$. We write $q \xrightarrow[M]{\gamma(a)} q'$ for $(q,\gamma,a,q') \in \delta$. We omit the subscript M when in no danger of confusion. Thus, the transition relation can be construed as a finite set of transition rules as above.

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 configuration 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]

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

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

- 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 [6] we dispense here with accepting and rejecting states: when no transition applies to a universal configuration then it has no children, and so the condition for acceptance is satisfied vacuously. Dually, a dead-end existential configuration is rejecting.

2.2 Acceptance and rejection

The *computation tree of M for configuration 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 $c \underset{\exists}{\longrightarrow} c'$ when $c \Rightarrow c'$ and c is existential, $c \underset{\exists}{\twoheadrightarrow} c'$ if $c \underset{\exists}{\longrightarrow}^* c'$ and c' is universal. (As usual, $\underset{\exists}{\longrightarrow}^*$ is the reflexive and transitive closure of $\underset{\exists}{\longrightarrow}$‘.) 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 $c \underset{\forall}{\longrightarrow} c'$ and $c \underset{\forall}{\twoheadrightarrow} c'$ are similar. We call configurations c' as above, for either $\underset{\exists}{\twoheadrightarrow}$ or $\underset{\forall}{\twoheadrightarrow}$, *alternation-pivots (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 $c \underset{\exists}{\twoheadrightarrow} c'$, then $c \in AC$.
2. If c is universal and $c' \in AC$ for all c' such that $c \underset{\forall}{\twoheadrightarrow} c'$, 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 $c \underset{\exists}{\twoheadrightarrow} c'$, then $c \in S$.
2. If c is universal and $c' \in S$ for all c' such that $c \underset{\forall}{\twoheadrightarrow} c'$, then $c \in S$.

Thus, AC is generated by the closure conditions $CC[AC]$. Note that $CC[S]$ is a Π^0_2 formula. For instance, (2) can be expressed as

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

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:

[3]We write ε for the empty string.

1. If c is existential and $c' \in RC$ for all c' such that $c \underset{\exists}{\twoheadrightarrow} c'$, then $c \in RC$.

2. If c is universal and $c' \in RC$ for some c' such that $c \underset{\forall}{\twoheadrightarrow} c'$, then $c \in RC$.

Again, RC is explicitly definable by a Π^1_1 formula.

We call a state a *dead-end* if no transition rule applies to it. We dub a universal dead-end state 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 configuration for w is in the set AC of accepted configurations, as defined above. Dually, *M rejects w* if that configuration 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

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

and the *language rejected by M* is

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

It is easy to see that $\mathscr{L}(M) \cap \overline{\mathscr{L}}(M) = \emptyset$. 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 **1.** *Let M be an ATM, T a computation tree of M for input w. If c is a configuration in the tree, with children $c_1 \ldots c_m$, 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. $c \underset{\exists}{\twoheadrightarrow} c'$ for some accept-state c', then $c_i \underset{\exists}{\twoheadrightarrow}^* c'$ 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 $c \underset{\exists}{\twoheadrightarrow} c'$, and so c is accepted, by definition of acceptance. If c_i is existential, then there must be an accepted c' such that $c_i \underset{\exists}{\twoheadrightarrow}^* c'$; but then $c \underset{\exists}{\twoheadrightarrow}^* c'$, so c is accepted.

Other cases are proved similarly. □

2.3 Divergence and totality

Note that if M has only existential states, and has for a given input only infinite computation traces, then that input is rejected: since there are no alternation-pivots, it is vacuously true that all alternation pivots are rejected. Still, an ATM may well neither accept nor reject an input string w. For example, if the computation tree of M for a given input 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, as is clear from the second-order definitions of acceptance and rejection. Indeed, the empty set satisfies the closure conditions for acceptance, and also 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.** *If an ATM is alternation well-founded then it is total.*

Proof. We prove the contra-positive: if a configuration 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 $c \underset{\forall}{\twoheadrightarrow} c'$ 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 $c \underset{\forall}{\twoheadrightarrow} c'$ for some alternation-pivot c' which is neither accepted nor rejected.

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

The converse of Proposition 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 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 **3.** *Let $\bar{M}$ be the dual of M. Then $\mathscr{L}(\bar{M}) = \bar{\mathscr{L}}(M)$, and so $\bar{\mathscr{L}}(\bar{M}) = \mathscr{L}(M)$.* □

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

PROPOSITION **4.** *For every machine M there are one-sided machines M^+ and M^- such that $\mathscr{L}(M) = \mathscr{L}(M^+)$, and $\overline{\mathscr{L}}(M) = \overline{\mathscr{L}}(M^-)$.*

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 configuration, where that alternation-pivot is neither accepted nor rejected. Thus, a set of configurations is closed under the closure conditions for AC in M iff it is closed under those conditions in M^+. So the two machines accept the same language. 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. For Σ_1 machines we further posit that the universal states have no applicable transitions, so they are accept-states without further ado. This is no loss of generality, since a Σ^0_1 ATM that does not satisfy this condition can be converted to an equivalent one (i.e. accepting the same language) which does, simply by removing all transitions that apply to universal states.

The definition of Π_k machines is the same, but with the initial state universal. Here again we posit that the existential states of Π_1 machines have no applicable transition rules.

THEOREM **5.** *Let $k \geq 1$. A language is Σ^0_k (Π^0_k) iff it is accepted by a Σ_k (Π_k, respectively) ATM.*

Proof. The proof is by induction on k. For the base case for Σ^0_1, let L be a language defined by a Σ^0_1 formula, that is

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

where

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

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

Conversely, if $L = \mathscr{L}(M)$ where M is a Σ_1 machine, then L is definable by a Σ^0_1 formula that states, for input w, the existence of a finite computation tree for w.

For the base case Π^0_1, suppose L is defined by a Π^0_1 formula

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

Define a Π_1 machine M that accepts L, as follows. M generates strings $w_1\#\cdots\#w_r$ 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 $w_1\#\cdots\#w_r$.

Conversely, if $L = \mathscr{L}(M)$ 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 on k generalizes the induction basis, in referring to a sub-computation for $k-1$, using IH, rather than to a deterministic sub-computation. □

3 Alternation and the analytical hierarchy

3.1 Accepted languages are inductive (Π^1_1)

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. We write φ_0 for bounded formulas, i.e. with all quantifiers restricted to substrings of a given string.

PROPOSITION **6.** *The following conditions are equivalent for a language $L \subseteq \Sigma^*$.*

I1 *L is defined by a formula of the form $\varphi[w] \equiv \forall f\ \varphi_0[w, f]$, where f is ranging over $\Sigma^* \to \Sigma^*$.*

I2 *L is defined by a formula of the form $\forall f\ \exists x\ \varphi_0[w, f, x]$.*

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 [9]. 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. □Note that the use of a set quantifier in I4 implies the need to have an alternation of first-order quantifiers, 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 powerful than Σ^0_1 [7, 8].

A language $L \subseteq \Sigma^*$ is *inductive* (Π^1_1) when it satisfies the equivalent conditions of Proposition 6 (see e.g. [4]).

Recall that our definition above of acceptance by an ATM refer to the set AC of accepted configurations, which is Π^1_1 definable. We therefore have:

PROPOSITION **7.** *Every language accepted by an ATM is inductive.*

3.2 Inductive languages are accepted

PROPOSITION **8.** *Every inductive language is accepted by an ATM.*

Proof. We refer to characterization (I3) of Π^1_1 languages. 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 infinite formula

$$\begin{array}{l} \forall z_0\ \ (\exists x\, \varphi_0[w, z_0, x]) \\ \qquad \vee\ \forall z_1\ \ (\exists x\, \varphi_0[w, z_0\,\$\,z_1, x]) \\ \qquad\qquad \vee\ \forall z_2\ \ (\exists x\, \varphi_0[w, z_0\,\$\,z_1\,\$\,z_2, x]) \\ \qquad\qquad\qquad \vee\ \forall z_3\ \ (\exists x\, \varphi_0[w, z_0\,\$\,z_1\,\$\,z_2\,\$\,z_3, x]) \\ \qquad\qquad\qquad\qquad\qquad\quad \vee\ \cdots \end{array} \tag{1}$$

We use here infinitary formulas for informal expository purpose; compare [10, 11].

Formula (1) is captured by an ATM which, on input w,

1. chooses by universal nondeterminism a value z_0;[4]

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

2. for each such choice for z_0, branches by existential nondeterminism to

   (a) guess (by existential nondeterminism) an x, then attempt to verify (deterministically) that $\varphi_0[w,z_0,x]$;

   (b) proceed to choose by universal nondeterminism a z_1;

   (c) etc.

□

Combining Propositions 7 and 8 we conclude:

THEOREM **9.** *L is inductive iff it is accepted by an ATM.*

3.3 Total machines and hyper-arithmetical languages

Recall that a language is Δ^1_1 when it is both Π^1_1 and Σ^1_1.

THEOREM **10.** *A language is Δ^1_1 iff it is accepted by a total ATM.*

Proof. If a language L is accepted by a total ATM M, then it is inductive, by Theorem 9. Also, by Proposition 3, $\bar{M}$ accepts $\bar{L}$, which is therefore inductive as well.

For the converse, assume that $L = \mathscr{L}(M_0)$ and $\bar{L} = \mathscr{L}(M_1)$. By Proposition 4 we may assume that neither machine has rejected configurations. Thus $\bar{L}$ is rejected by the machine $\bar{M}_1$, which has no accepted configuration.

We wish to construct out of M_0 and $\bar{M}_1$ a total machine M that accepts L, and thus rejects $\bar{L}$. An initial idea is to emulate the classical proof that a language which is both semi-decidable (i.e. RE) and co-semi-decidable must be decidable. This would consist in combining the two machines, by constructing a two-tape machine whose states are are tuples $\langle q_0,q_1,j\rangle$, with q_0 a state of M_0 and q_1 a state of $\bar{M}_1$, and where $j \in \{0,1\}$ indicates which machine moves next. The type of $\langle q_0,q_1,j\rangle$ is the type of q_j, and the states take turns in operating on their respective tapes. Acceptance of a configuration is determined (or so it seems...) by the M_0 portion of the combined machine, since $\bar{M}_1$ has no accepted configuration, and rejection is determined by the $\bar{M}_1$ portion, since M_0 has no rejected configurations. The initial state of the combined machine M, say s, starts a preliminary computation phase, which copies the input, given say on M_0's tape, to the tape associated with $\bar{M}_1$. This phase leads to the state $\langle s_0,s_1,0\rangle$, where s_0 and s_1 are the initial states of M_0 and $\bar{M}_1$, respectively.

One problem with that tentative construction is that the acceptance semantics of each portion is in fact impacted by the other, since the latter may introduce alternation-pivots. Consider, for example, a universal configuration c of M_0 which is accepted in M_0, because all its alternation-pivots in M_0 are accepted. When c is paired with a configuration of $\bar{M}_1$ we get a compound configuration which might have, via the operation of $\bar{M}_1$, additional alternation-pivots whose "M_0-portion" is not accepted in M_0. Thus the combined configuration will not be accepted in the combined machine, even though c is accepted in M. A dual problem might occur with existential configurations of $\bar{M}_1$.

We address the issue above by refining the construction of the combined machine M. We also include in M the states of M_0 and of $\bar{M}_1$ as given (assuming of course that the two machines have no state in common). A state q_0 of M_0 behaves in M exactly as in M_0, but disregarding the tape associated with $\bar{M}_1$; and similarly for states of $\bar{M}_1$.

A state $\langle q_0,q_1,0\rangle$ which is universal (i.e. where q_0 is universal in M_0), rather than transitioning (for a given tape-values) to a configuration with a state of the form $\langle p_0,q_1,1\rangle$, is first switching to an existential state $\langle q'_0,q_1,0\rangle$ (with q'_0 a fresh auxiliary state), branching then to $\langle p_0,q_1,1\rangle$ and to the state p_0 itself.

Thus we create an alternation pivot above the current configuration, which is accepted iff the 0-part of the current configuration is accepted by M_0.

We refine dually the behavior of M for existential states $\langle q_0, q_1, 1\rangle$.

It is easy to see that in the revised machine M a configuration with state $\langle q_0, q_1, j\rangle$ and tape-values $\langle w_0, w_1\rangle$ is accepted with $j = 0$ iff the configuration (q_0, w_0) is accepted in M_0, and is rejected with $j = 1$ iff (q_1, w_1) is rejected in $\bar{M}_1$. Thus $\langle s_0, s_1, 0\rangle$ is accepted with tape-values $\langle w, w\rangle$ iff $w \in L$, and rejected iff $w \in \bar{L}$; it follows that M (with initial configuration $(s, \langle w, \varepsilon\rangle)$) is a total ATM that accepts L. □

We are not aware of proofs of similar results that do not refer to a calibration of transfinite computing via Kleene's set $\mathscr{O}$ of recursive ordinals. This is the case, in particular, for a theorem analogous to the above, in [3], for a programming languages IND with random assignments. In contrast, our proof above does not address infinitely-branching computation, and consequently does not depend on transfinite stage-comparison methods.

3.4 The Analytical Hierarchy

We further generalize ATMs with negation gates. That is, in addition to existential and universal states such machines may have *negation* states, with a unique transition for each symbol scanned.

Negation-gates are traditionally defined locally, by the boolean condition: a configuration with a negation state is accepted (rejected) when its child-configuration is rejected (accepted, respectively). Such states can be dispensed with as in [2], using a duality construction. We consider instead negation gates with a stronger semantics: a negation configuration is accepted (rejected) when its child-configuration is not rejected (not accepted, respectively). Thus, our semantics is based on asserting acceptance and rejection on a "no contest" basis, and referring to the behavior of the computation tree as a whole, much as we have done for the semantics of universal nondeterminism.

Contrary to the definition of acceptance and rejection so far, our semantics of negation is not monotonic, because the closure condition defining negation refers to the absence of acceptance, a condition that may turn from true to false as the set of accepted configuration is generated. We therefore restrict attention to computation trees with a finite bound k on the number of negation configuration permissible along each branch. For such computation trees we can define the semantics of negation by induction on k.

From Theorem 9 we obtain that every Σ^1_1 language is accepted by an ATM with a single negation state. Using such a machine as oracle to an ATM without negation, we obtain ATMs (with at most one negation gate along every computation trace) that accepts the Π^1_2 languages. More generally, we have

THEOREM **11.** *The languages accepted by ATMs with negation, using $< k$ negation gates along every computation trace, are precisely the Π^1_k languages of the analytical hierarchy.*

4 Conclusion

The combined use of existential and universal nondeterminism has been of interest primarily in Computational Complexity theory, but has not been viewed 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. This 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^0_1, 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 alternation for arbitrary computations with global semantics, which yields Π^1_1 as a limit of the arithmetical hierarchy (starting with Σ^0_1).

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. These might provide insights and machine-based proofs for Higher Recursion Theory. A potential example is the Spector-Gandy-Kleene Theorem, stating that a language is inductive iff it has a definition of the form $(\exists S \in \Delta^1_1)\ (\forall w)\ \varphi[S,w,x]$, with φ recursive. We shall return to this connection elsewhere.

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] Phokion Kolaitis. Game quantification. In *Model-Theoretic Logics*, pages 365–421. Springer-Verlag, New York, 1985.

[6] Dexter Kozen. *Theory of Computation*. Springer, London, 2006.

[7] G. Kreisel. La prédicativité. *Bull. Soc. math. France*, 88:371–391, 1960.

[8] Georg Kreisel. Survey of proof theory. *Journal of symbolic Logic*, 33:321–388, 1968.

[9] Kazimierz Kuratowski and Alfred Tarski. Les opérations logiques et les ensembles projectifs. *Fund. Math.*, 17:240–248, 1931.

[10] Y. Moschovakis. *Elementary Induction on Abstract Structures*. North-Holland, Amsterdam, 1974.

[11] Yianis Moschovakis. The game quantifier. *Proc. AMS*, 31:245–250, 1971.

[12] H. Rogers. *Theory of Recursive Functions and Effective Computability*. McGraw-Hill, New York, 1967.