Borel Coalgebra and Non-Wellfounded Logic

Dexter Kozen[1] **and Francisco Mota**[1]

1 **Department of Computer Science**
Cornell University
Ithaca, New York 14853-7501, USA
{kozen,fmota}@cs.cornell.edu

Abstract

We introduce *Borel coalgebras* and *Borel automata* as a computational approach to basic descriptive set theory. We show that over any Polish space, Borel automata accept exactly the coanalytic sets, and total Borel automata (those that halt on all inputs) accept exactly the Borel sets. The latter result is a computational version of the Kleene–Suslin theorem. The ordinal rank of a Borel set is characterized as the running time of a Borel automaton. We show how these ideas lead to a general notion of *non-wellfounded logic* in which syntactic objects such as terms and formulas are elements of a final coalgebra. We relate these notions to the categorical theory of recursion schemes (Adamek, Milius, and Velebil 2006, Milius and Moss 2006) to provide a foundation for non-wellfounded logic.

Keywords and phrases coanalytic sets, Borel sets, coalgebra, non-wellfounded logic, IND programs

Digital Object Identifier 10.4230/LIPIcs...

1 Introduction

Mathematical logic has had a profound influence on the theory of computation for over half a century. From type theory to denotational semantics, from verification to model checking, from automated deduction to the Curry–Howard correspondence, the foundations of computation owe a great debt to logic.

In the reverse direction, the contributions of computation to the foundations of logic are perhaps less well established. However, one distinguishing characteristic of computation that can provide a fresh perspective is its dynamic nature, in contrast to traditional static nature of logic. In logic, models, valuations of variables, and truth values of predicates are regarded as fixed and immutable. Computation, on the other hand, often involves explicit operators such as assignments $x := e$ that can change state. In addition, programming languages often provide mutable data structures for maintaining information during a computation, which are typically absent from traditional treatments of logic. This dichotomy is reflected in the relationship between denotational and operational models in programming language semantics.

Although the static approach is perhaps more amenable to mathematical treatment, the dynamic perpective can lend insight and reinforce intuition. For example, in the realm of inductive definability, *Kleene's theorem* states that over $\mathbb{N}$, the inductive and Π^1_1 relations coincide, as do the hyperelementary and Δ^1_1 relations. However, this theorem has a more computational interpretation than is apparent from the standard development [5, 15]. In [6], a programming language IND was defined, and it was shown that over any structure, loop-free IND programs (respectively, IND programs that always halt, general IND programs) compute exactly the first-order (respectively, hyperelementary, inductive) relations. By Kleene's theorem, IND programs compute exactly the Π^1_1 relations over $\mathbb{N}$. Moreover, IND programs

can be used to give a more computationally motivated proof of Kleene's theorem. In [10], it was shown that IND programs with dictionaries, a common abstract data structure allowing storage and retrieval of data indexed by keys, compute exactly the Π^1_1 relations over any countable structure. This was not a new result, but rather a more computational perspective on a result of Barwise, Gandy, and Moschovakis [4, 14] (see [5, Corollary VI.3.9(i), p. 214]) stating that over any countable structure $\mathfrak{A}$, the Π^1_1 relations are equivalent to a certain class of inductively definable relations over $\mathbb{HF}\mathfrak{A}$, where $\mathbb{HF}\mathfrak{A}$ refers to the structure $\mathfrak{A}$ augmented with its hereditarily finite sets. But the result of [10] clarifies the role of the hereditarily finite sets: like dictionaries, they are simply a data structure, nothing more nor less. The difference is that IND programs and dictionaries are more programmable.

Our purpose in this paper is to further this agenda. In section §2 we introduce *Borel coalgebras* and *Borel automata* as a model of computation for describing constructs from basic descriptive set theory [11, 17], but with a decidedly more computational flavor. We show that over any Polish space, Borel automata accept exactly the coanalytic sets, and total Borel automata (those that halt on all inputs) accept exactly the Borel sets. The latter result is a computational version of the Kleene–Suslin theorem [16]. The ordinal rank of a Borel set is characterized as the running time of a Borel automaton. These running times can be any countable ordinal, but Borel automata that accept their inputs within a uniform countable time bound accept only Borel sets; this is shown by "clocking" the computation. The semantics is a direct generalization of the semantics of alternating Turing machines and IND programs. The main contributions here are not so much the results themselves, but rather a more computational perspective on the basic notions of descriptive set theory [11, 16, 17].

In §4 we impose two natural restrictions on Borel coalgebras, the *finitary* and *uniform* Borel coalgebras, and characterize the expressive power of these restricted models. The latter turn out to be equivalent to programs of the IND programming language introduced in [6] interpreted over countable structures. This is the "Kleene" part of the Kleene–Suslin theorem.

Finally, in §5 we discuss a larger context for the dynamic approach based on a non-wellfounded version of infinitary logic [7] that is also related to coalgebraic logic [18] and the categorical theory of recursion schemes [3, 12, 13]. Normally syntactic objects such as terms and formulas are defined inductively, viewing the nonlogical and logical symbols as algebraic constructors. This is so with classical first-order logic and the infinitary logics $L_{\alpha\beta}$, notably $L_{\omega_1\omega}$ [7], which are exclusively wellfounded. In our approach, syntactic objects are *coterms* over an algebraic signature, which are elements of a final coalgebra for the signature. As coterms are not wellfounded in general, we must accommodate nonhalting; this is accomplished with the use of three-valued (Łukasiewicz) logic, the third value $\bot$ representing "don't know yet (and maybe never will)."

We emphasize that although our treatment involves coalgebras and coterms, the logic is algebraic, not coalgebraic. We elaborate on this point in §5, in which we give a general account of non-wellfounded logic, including notions of interpretation and substitution. Our main result of this section is a free construction that for any F-algebra A and interpretation of variables over A gives a canonical interpretation of F-coterms in the flat Scott domain $A_\bot$, using results from the categorical theory of recursion schemes [3, 12, 13]. Since the F-coterms form the final F-coalgebra, this interpretation composes with the unique coalgebra morphism from any F-coalgebra S to provide a computational semantics for S when viewed as a program.

s : all A	where	$A \in \wp_{\omega_1}(S)$
s : some A	where	$A \in \wp_{\omega_1}(S)$
s : not t	where	$t \in S$
s : if φ then t else u	where	$\varphi \in \Phi$ and $t, u \in S$

Figure 1 Borel Coalgebra

2 Borel Coalgebras and Borel Automata

▶ **Definition 2.1.** Fix a set Φ of basic predicates in some language. A *Borel coalgebra* over Φ is a possibly infinite program made of labeled statements of the form shown in Figure 1, where S is a set of statement labels or *states* and $\wp_{\omega_1}(S)$ denotes the countable powerset of S. Formally, a Borel coalgebra is a pair (S, Δ) where S is a set of states and Δ is a structure map

$$\Delta : S \to \wp_{\omega_1}(S) + \wp_{\omega_1}(S) + S + (\Phi \times S^2),$$

where the components correspond to all A, some A, not t, and if φ then t else u from left to right. Alternatively, Δ comprises two maps ℓ and δ, where $\delta(s)$ is the set of states on which s depends and $\ell(s)$ tells us what kind of state s is, as given in the following table.

Program Notation	Type of $\Delta(s)$	$\delta(s)$	$\ell(s)$
s : all A	$\wp_{\omega_1}(S)$	A	all
s : some A	$\wp_{\omega_1}(S)$	A	some
s : not t	S	t	not
s : if φ then t else u	$\Phi \times S^2$	(t, u)	$\varphi \in \Phi$

▶ **Definition 2.2.** A *Borel automaton* is a triple (S, Δ, s_0) where (S, Δ) is a Borel coalgebra and $s_0 \in S$ is the *starting state* of the automaton.

2.1 Intuitive Operation

The input to a Borel automaton is a mathematical structure $\mathfrak{A}$ in the language of Φ and a valuation σ interpreting variables in Φ as elements of $\mathfrak{A}$. The computation consists of processes generating a computation tree downward from the start state s_0 and passing acceptance or rejection information upward from the leaves to the root.

We start with a single process p_0 in state s_0. During the computation, if a process p is in a non-leaf state s, that is, a state for which $\delta(s) \neq \varnothing$, then p spawns a set of child processes, one for each element of $\delta(s)$. The process p then suspends, waiting for reports of acceptance or rejection from its children. The child process associated with $t \in \delta(s)$ proceeds similarly from state t.

When enough information is received from its children, the process p waiting at a state s accepts or rejects as appropriate and passes this information back up to its parent. A process waiting at a "some" state accepts if at least one of its children accepts and rejects if all of its children reject. A process waiting at an "all" state accepts if all of its children accept and rejects if at least one of its children rejects. A process waiting at a "not" state rejects if its unique child process accepts and accepts if its unique child rejects. A process waiting at an "if φ then t else u" state depends on the truth of φ: if $\mathfrak{A}, \sigma \models \varphi$ then this

process accepts if the child process at t accepts and rejects if the process at t rejects; if $\mathfrak{A}, \sigma \models \neg\varphi$ then the process accepts if the process at u accepts and rejects if the process at u rejects.

The computation is said to *accept* if the original process p_0 waiting at the start state s_0 accepts. It is said to *reject* if p_0 rejects. It is said to *halt* if p_0 either accepts or rejects.

Because of countable branching, a process waiting at a state for reports from its children can take more than finite time to accept or reject. Thus the notion of *time* must be suitably generalized. The "time" at which a process halts can be any countable ordinal.

2.2 Formal Semantics

To execute a Borel automaton, we must fix a structure $\mathfrak{A}$ and a valuation σ and then simulate the processes as described above. There is no imposition of well-foundedness on the transition structure, so a process may depend on its own value, directly or indirectly. For example, the circular statements "$s : \mathsf{all}\ \{s\}$" and "$s_1 : \mathsf{not}\, s_2\ ;\ s_2 : \mathsf{not}\, s_1$" are allowed. We thus interpret the value of each process by successive approximation, using three-valued logic with truth values $\mathsf{3} = \{0, 1, \bot\}$ where $\bot$ represents "don't know yet (and maybe never will)." There are two natural partial orders defined on $\mathsf{3}$, namely the *Łukasiewicz order* $\leq$ and the *Scott (information) order* $\sqsubseteq$, defined by the following Hasse diagrams:

Łukasiewicz order $\leq$ Scott (information) order $\sqsubseteq$

As algebraic constructors on $\mathsf{3}$, the operators $\vee$ and $\wedge$ give the supremum and infimum, respectively, in the Łukasiewicz order $\leq$, and the operator $\neg$ inverts the order, i.e. maps 0 to 1, 1 to 0, and $\bot$ to $\bot$.

The computation of a Borel automaton (S, Δ, s_0) on input σ is defined formally in terms of an inductively defined labeling $L_\sigma : S \to \mathsf{3}$. Given any labeling $L : S \to 3$, we define a new labeling $\tau_\sigma(L)$ as follows:

$$\tau_\sigma(L)(s) = \begin{cases} \bigwedge_{t\in A} L(t), & \text{if } \Delta(s) = (\mathsf{all}, A), \\ \bigvee_{t\in A} L(t), & \text{if } \Delta(s) = (\mathsf{some}, A), \\ \neg L(t), & \text{if } \Delta(s) = (\mathsf{not}, t), \\ L(t), & \text{if } \Delta(s) = (\varphi, t, u) \text{ and } \mathfrak{A}, \sigma \models \varphi, \\ L(u), & \text{if } \Delta(s) = (\varphi, t, u) \text{ and } \mathfrak{A}, \sigma \models \neg\varphi. \end{cases}$$

The map τ_σ is monotone in the information order $\sqsubseteq$ defined pointwise on labelings. By the Knaster–Tarski theorem, τ_σ has a $\sqsubseteq$-least fixpoint $L_\sigma : S \to \mathsf{3}$. This is the least L such that $\tau_\sigma(L) \sqsubseteq L$. The least fixpoint can also be obtained as a limit of approximants L_α:

$$L_0(s) = \bot \qquad L_{\alpha+1} = \tau(L_\alpha) \qquad L_\lambda = \bigsqcup_{\alpha<\lambda} L_\alpha \qquad L_\sigma = \bigsqcup_{\alpha} L_\alpha.$$

Here λ denotes a limit ordinal and $\bigsqcup$ the supremum in the information order. If S is countable, the closure ordinal of the inductive definition of L_σ is countable. For the final coalgebra, the closure ordinal is ω_1. We say that S *accepts* if $L_\sigma(s_0) = 1$ and *rejects* if $L_\sigma(s_0) = 0$. We say that the automaton *halts* if it either accepts or rejects.

2.3 Time

The *halting time* $T_\sigma(s)$ of a state $s \in S$ on input σ is the least ordinal α such that $L_\alpha(s) \neq \bot$, if such an α exists. If $T_\sigma(s)$ exists, then it is countable. For $a \in \{0, 1, \bot\}$, let $E_a(s) = \{t \in \delta(s) \mid L_\sigma(t) = a\}$.

$$T_\sigma(s) = \begin{cases} \inf_{t \in E_1(s)} T_\sigma(t) + 1, & \ell(s) = \mathsf{some} \text{ and } E_1(s) \neq \varnothing, \\ \sup_{t \in E_0(s)} T_\sigma(t) + 1, & \ell(s) = \mathsf{some} \text{ and } E_0(s) = \delta(s), \\ \sup_{t \in E_1(s)} T_\sigma(t) + 1, & \ell(s) = \mathsf{all} \text{ and } E_1(s) = \delta(s), \\ \inf_{t \in E_0(s)} T_\sigma(t) + 1, & \ell(s) = \mathsf{all} \text{ and } E_0(s) \neq \varnothing, \\ T_\sigma(t) + 1, & \ell(s) = \mathsf{not} \text{ and } \delta(s) = \{t\}, \\ T_\sigma(t) + 1, & \Delta(s) = (\varphi, t, u) \text{ and } \mathfrak{A}, \sigma \models \varphi, \\ T_\sigma(u) + 1, & \Delta(s) = (\varphi, t, u) \text{ and } \mathfrak{A}, \sigma \models \neg\varphi, \end{cases}$$

and undefined otherwise. We say that a Borel automaton S is *uniformly α-time bounded* for a countable ordinal α if, whenever S halts, it does so within time α; that is, $T_\sigma(s_0) < \alpha$ for any halting input σ. Note that every Borel automaton is ω_1-time bounded.

2.4 Useful Constructions

▶ **Definition 2.3.** An alternative formulation of Borel coalgebra replaces conditional statements "s : $\mathsf{if}\ \varphi\ \mathsf{then}\ t\ \mathsf{else}\ u$" with test statements "$s : \varphi$" which accept immediately if $\mathfrak{A}, \sigma \vDash \varphi$ and reject immediately if $\mathfrak{A}, \sigma \nvDash \varphi$. We call these *oblivious Borel coalgebras* because tests can only appear at the leaves of the computation tree, so the computation tree is the same for all inputs. *Non-oblivious Borel coalgebras* (as defined in the previous section) can simulate oblivious Borel coalgebras by replacing each test $s : \varphi$ with the statements:

$$s : \mathsf{if}\ \varphi\ \mathsf{then\ accept\ else\ reject} \qquad \mathsf{accept} : \mathsf{all}\ \varnothing \qquad \mathsf{reject} : \mathsf{some}\ \varnothing$$

where accept accepts immediately and reject rejects immediately. Likewise, non-oblivious Borel coalgebras can be made oblivious by replacing every conditional "$s : \mathsf{if}\ \varphi\ \mathsf{then}\ t\ \mathsf{else}\ u$" with the statements:

$$s : \mathsf{some}\,\{s_1, s_2\} \qquad s_1 : \mathsf{all}\,\{s_3, u\} \qquad s_2 : \mathsf{all}\,\{s_4, t\} \qquad s_3 : \mathsf{not}\, s_4 \qquad s_4 : \varphi$$

or more succinctly, allowing compound statements,

$$s : \mathsf{some}\,\{\mathsf{all}\,\{\mathsf{not}\,\varphi, u\}, \mathsf{all}\,\{\varphi, t\}\}$$

This says that working with oblivious or non-oblivious automata is mostly a matter of taste.

▶ **Definition 2.4.** A Borel coalgebra is *positive* if it has no negation statements. Due to DeMorgan's laws, we can make every Borel coalgebra positive by following a two-step process. Given S, build the dual machine with a disjoint copy of the statement labels $S' = \{s' \mid s \in S\}$.

if S has	then S' has
$s : \mathsf{all}\, A$	$s' : \mathsf{some}\, A'$
$s : \mathsf{some}\, A$	$s' : \mathsf{all}\, A'$
$s : \mathsf{not}\, t$	$s' : \mathsf{not}\, t'$
$s : \mathsf{if}\ \varphi\ \mathsf{then}\ u\ \mathsf{else}\ v$	$s' : \mathsf{if}\ \varphi\ \mathsf{then}\ u'\ \mathsf{else}\ v'$

Note that s accepts when s' rejects and vice versa. Now build a machine from the disjoint union $S \cup S'$ by replacing each negation:

if S, S' have	then $S \cup S'$ has
s : not t	s : all $\{t'\}$
s' : not t'	s' : all $\{t\}$

This construction does not preserve the obliviousness, and conversely the oblivious construction does not preserve positivity. To have both, we need the basic predicates to be closed under negation, allowing us to replace "s_3 : not s_4" with "s_3 : if $\neg\varphi$ then accept else reject" in the oblivious construction, thereby preserving positivity.

▶ **Definition 2.5.** Given two positive Borel coalgebras with disjoint state sets S_1, S_2, we build an *interleaved coalgebra* $S_1 \otimes S_2$ with states $(S_1 \times S_2) \cup (S_2 \times S_1)$ as follows:

if S_1, S_2 have	then $S_1 \otimes S_2$ has
s : all A	(s,t) : all $\{(t,r) \mid r \in A\}$
s : some A	(s,t) : some $\{(t,r) \mid r \in A\}$
s : if φ then u else v	(s,t) : if φ then (t,u) else (t,v)

This machine $S_1 \otimes S_2$ accepts if S_1 or S_2 accepts before the other rejects, and rejects if S_1 or S_2 rejects before the other accepts. Therefore if S_1 and S_2 agree whenever they both accept or reject, then $S_1 \otimes S_2$ agrees with both and halts whenever either of the two halt. By using the positive construction (Definition 2.4), we can interleave any two Borel coalgebra.

▶ **Lemma 2.6.** *If $T_\sigma(s) \leq T_\sigma(t)$, then $L_\sigma(s,t) = L_\sigma(s)$ and $T_\sigma(s,t) = T_\sigma(s) \cdot 2$.*

▶ **Definition 2.7.** For each countable ordinal $\alpha < \omega_1$ we construct a Borel automaton called the α-clock, with $S = \alpha + 1$ and $s_0 = \alpha$ and the single rule for all $\beta \in S$:

$$\beta : \text{some } \{\gamma : \gamma < \beta\}.$$

Regardless of input, this automaton runs for α steps and then rejects. By interleaving the α-clock with a Borel automaton S, we enforce a uniform time bound on S, as any input that takes longer than α steps to compute in the original automaton is automatically rejected. We call this process "clocking."

3 Characterization of Coanalytic and Borel Sets

Consider the class of Borel automata with a single variable x ranging over elements of a Polish space $\mathfrak{A}$ and atomic formulas $x \in A$, where A is a basic open set of $\mathfrak{A}$. The prototypical example of a Polish space is the Baire space ω^ω, in which case x ranges over ω^ω and A would be one of A_y for $y \in \omega^*$ representing the basic open set $\{a \in \omega^\omega \mid y \leq a\}$, where $\leq$ represents the prefix relation. For such automata, we say that S *accepts* $a \in \mathfrak{A}$ if $L_\sigma(s_0) = 1$, where $\sigma(x) = a$. We define

$$L(S) = \{a \in \mathfrak{A} \mid S \text{ accepts } a\},$$

the set of elements of $\mathfrak{A}$ accepted by S.

▶ **Theorem 3.1.** *Let A be a subset of a Polish space. A is coanalytic iff $A = L(S)$ for some Borel automaton S.*

Proof. ($\Leftarrow$) Without loss of generality assume S is oblivious. The acceptance condition is

$$\forall L \ (\tau_\sigma(L) \sqsubseteq L \Rightarrow L(s_0) = 1)\,, \tag{1}$$

where

$$\tau_\sigma(L)(s) = \begin{cases} \bigvee_{t\in\delta(s)} L(t), & \ell(s) = \mathsf{some}, \\ \bigwedge_{t\in\delta(s)} L(t), & \ell(s) = \mathsf{all}, \\ \neg L(\delta(s)), & \ell(s) = \mathsf{not}, \\ [\mathfrak{A}, \sigma \vDash \varphi], & \ell(s) = \varphi. \end{cases} \tag{2}$$

where $[\mathfrak{A}, \sigma \vDash \varphi] \in 2$ denotes the truth value of this basic predicate. Because the set of states reachable from the start state s_0 is countable and duplication of siblings does not affect acceptance or rejection, we can assume without loss of generality that $S = \omega^*$, $s_0 = \varepsilon$, and $\delta(s) = \{sn \mid n \in \omega\}$. By a long sequence of transformations (the full argument can be found in Appendix A), the acceptance condition (1) can be transformed to Π^1_1 form

$$\forall L \ \forall f \ \forall g \ \exists s \ \exists n \ (\psi(\sigma, L, s, f(s), g(s), n) \Rightarrow L(\varepsilon) = 1), \tag{3}$$

so the set

$$\begin{aligned} \{\sigma \mid S \text{ accepts } \sigma\} &= \{\sigma \mid \forall L \ (\tau_\sigma(L) \sqsubseteq L \Rightarrow L(\varepsilon) = 1)\} \\ &= \{\sigma \mid \forall L \ \forall f \ \forall g \ \exists s \ \exists n \ (\psi(\sigma, L, s, f(s), g(s), n) \Rightarrow L(\varepsilon) = 1)\} \end{aligned}$$

is coanalytic.

($\Rightarrow$) If A is coanalytic, it can be constructed via a Suslin scheme:

$$A = \bigcap_{x\in\omega^\omega} (M_\varepsilon \cup M_{x_0} \cup M_{x_0x_1} \cup M_{x_0x_1x_2} \cup \cdots) \tag{4}$$

where M_α is a family of basic open sets in $\mathfrak{A}$ indexed by finite sequences of natural numbers $\alpha \in \omega^*$. We build a Borel automaton with states ω^*, start state ε, and program:

$$\alpha : \mathsf{if}\ \sigma(x) \in M_\alpha\ \mathsf{then\ accept\ else\ all}\ \{\alpha n \mid n \in \omega\}. \tag{5}$$

The automaton checks that along all paths $x = x_0x_1\cdots$, eventually $\sigma(x) \in M_{x_0\cdots x_n}$. Then $L_\sigma(\varepsilon) = 1$ iff $\sigma(x) \in A$, so this Borel automaton accepts A. ◀

The following is our main theorem.

▶ **Theorem 3.2.** *Let A be a subset of a Polish space. The following are equivalent:*

1. *A is analytic and coanalytic.*
2. *A is Borel.*
3. *A is accepted by a well-founded Borel automaton. This means that all computation paths on any input are finite.*
4. *A is accepted by a Borel automaton with uniform time bound $\alpha < \omega_1$.*
5. *A is accepted by a total Borel automaton, i.e. one that halts on all inputs.*

▶ Remark. The equivalence ($1 \Leftrightarrow 2$) is known as the Kleene–Suslin theorem (see e.g. [11]). We give here a more computationally flavored proof.

Proof. ($2 \Leftrightarrow 3$) A well-founded Borel automaton in oblivious mode is an infinitary term representing the inductive construction of the set A as a Borel set.

($3 \Rightarrow 4$) If the computation is oblivious, the computation tree does not depend on the input σ. Since the tree is well founded, its ordinal is countable, and this is a uniform time bound for all inputs.

($4 \Rightarrow 3$) Given any Borel automaton with a uniform time bound α, we can interleave the automaton with the α-clock, according to Definition 2.7. The clocked automaton is well founded since the ordinal α is; any computation that takes too long is now forcibly truncated. The language A is preserved because all the inputs were originally accepted in fewer than α steps.

($3 \Rightarrow 5$) Every computation path of a well-founded automaton is finite. It is easily shown by well-founded induction on the structure of the automaton that on any input, every node is eventually labeled with either 0 or 1.

($5 \Rightarrow 1$) By Theorem 3.1, A is coanalytic. Moreover, because the automaton halts on all inputs, the dual automaton (Definition 2.4) accepts the complement of A, so the complement of A is also coanalytic, and therefore A is analytic.

($1 \Rightarrow 4$) This is the most interesting case. This can be shown by a product construction analogous to the proof that an r.e., co-r.e. set is recursive or an inductive, coinductive set (as represented by complementary IND programs) is hyperelementary. If A and $\sim A$ are both coanalytic, they have Suslin schemes (4) defined by basic open sets M_α and N_β, respectively. Moreover, if the space is Polish (completely metrizable and separable), we can arrange that the diameters of the sets $\sim M_\alpha$ along any path decrease to 0, and similarly for $\sim N_\beta$. Then A and $\sim A$ are accepted by automata of the form (5):

$$\alpha : \textsf{if } \sigma(x) \in M_\alpha \textsf{ then accept else all } \{\alpha n \mid n \in \omega\}$$
$$\beta : \textsf{if } \sigma(x) \in N_\beta \textsf{ then accept else all } \{\beta n \mid n \in \omega\},$$

respectively. These automata have only choice nodes and universal branching, thus they never reject. For any input, exactly one machine will accept. We construct a product machine that runs both machines together, alternating steps between the two.

$$(\alpha, \beta, 0) : \textsf{if } \sigma(x) \in M_\alpha \textsf{ then accept else all } \{(\alpha n, \beta, 1) \mid n \in \omega\}$$
$$(\alpha, \beta, 1) : \textsf{if } \sigma(x) \in N_\beta \textsf{ then accept else all } \{(\alpha, \beta n, 0) \mid n \in \omega\}.$$

The start state is $(\varepsilon, \varepsilon, 0)$.

We claim that no input ever follows an infinite path in the computation tree all the way down. To do so, it would have to fail all infinitely many membership tests along the path. These tests are determined by an interleaved pair of infinite strings $x_0y_0x_1y_1x_2y_2\cdots$ such that the tests alternate between M and N; that is, the tests are M_ε, N_ε, M_{x_0}, N_{y_0}, $M_{x_0x_1}$, $N_{y_0y_1}, \ldots$. But for $\sigma(x)$ to fail all these tests would imply

$$\sigma(x) \notin \bigcap_{x \in \omega^\omega} (M_\varepsilon \cup M_{x_0} \cup M_{x_0x_1} \cup \cdots) \cup \bigcap_{y \in \omega^\omega} (N_\varepsilon \cup N_{y_0} \cup N_{y_0y_1} \cup \cdots) = A \cup \sim A,$$

a contradiction. Since the $\sim M_\alpha$ and $\sim N_\beta$ are closed and their diameters decrease to 0 along any path, there is no sequence of inputs that follow the path arbitrarily far down either, because it would be a Cauchy sequence converging to an input that follows the path all the way down. Thus the computation tree can be pruned to give a well-founded automaton accepting all inputs. Let $\alpha < \omega_1$ be the ordinal of this tree.

Now for any $\sigma(x) \in A$, since $\sigma(x) \notin {\sim}A$, there exists $y \in \omega^\omega$ such that

$$\sigma(x) \notin N_\varepsilon \cup N_{y_0} \cup N_{y_0 y_1} \cup N_{y_0 y_1 y_2} \cup \cdots .$$

Pruning the computation tree of all paths $x_0 y_0' x_1 y_1' x_2 y_2' \cdots$ such that $y' \neq y$ yields a tree containing an accepting computation tree of the original automaton for A on input $\sigma(x)$, and the ordinal of this tree is bounded by α. Since $\sigma(x) \in A$ was arbitrary, the running time of the original automaton for A is uniformly bounded by α. ◀

4 Definability with Restricted Borel Coalgebras

The previous section showed that Borel coalgebras are very powerful, accepting precisely the coanalytic sets and deciding precisely the Borel sets, when interpreted over a Polish space. In this section we consider two natural restrictions to Borel coalgebras and consider analogs of Theorems 3.1 and 3.2 for these restrictions.

4.1 Finitary Borel Coalgebras

Under the process interpretation of a Borel coalgebra (§2.1), we assume that a process can recognize when all of its (infinitely many) children have made a decision. This is unrealistic for real-world processes operating in finite time, so in this section we limit ourselves to finitely branching processes.

▶ **Definition 4.1.** A Borel coalgebra (S, Δ) is *finitary* if $\delta(s)$ is finite for all $s \in S$. A Borel automaton (S, Δ, s_0) is *finitary* if (S, Δ) is finitary.

▶ **Lemma 4.2.** *If S is a finitary Borel coalgebra, then L_ω is a fixpoint of τ_σ.*

Proof. Suppose $\tau_\sigma(L_\omega)(s) = b$ for some $b \in \{0, 1\}$. Because $\delta(s)$ is finite and $L_\omega = \bigsqcup_{n\in\omega} L_n$ it follows that there exists some $n < \omega$ such that $L_n(t) = L_\omega(t)$ for all $t \in \delta(s)$. Therefore $L_{n+1}(s) = b$ and thus $L_\omega(s) = b$. Therefore L_ω is a fixed point of τ. ◀

▶ **Theorem 4.3.** *If $\mathfrak{A}$ is a topological space and the basic predicates Φ are of the form "$x \in A$" where A is a clopen subset of $\mathfrak{A}$, then $L(S)$ is open in $\mathfrak{A}$ for any finitary Borel automaton S. Conversely if $\mathfrak{A}$ has a countable clopen basis and $A \subseteq \mathfrak{A}$ is open, there exists a finitary Borel automaton that accepts A.*

Proof. The full proof is given in Appendix A. ◀

▶ **Theorem 4.4.** *Let $\mathfrak{A}$ be the Cantor space 2^ω. If S is a finitary Borel automaton over the language of clopen subsets of $\mathfrak{A}$, and S halts on all inputs, then there is a finite time bound $n < \omega$ such that S halts in n steps for all inputs.*

Proof. This is a straightforward compactness argument. The proof details are given in Appendix A. ◀

4.2 Uniform Borel Coalgebras

Borel coalgebras allow for arbitrarily complex branching behavior, encoding intractable or uncomputable functions through the transition function $\delta(s)$. In this section we look at a restriction of Borel coalgebras where the transition structure has a finite description, and we show that this corresponds to the IND programming language.

Let Φ be a set of basic predicates that is closed under substitution of free variables with natural numbers. For example, in the first-order language of rings, every free variable x in a basic predicate φ can be substituted with a number $n \in \mathbb{N}$ yielding the basic predicate $\varphi[n/x]$, where n represents the term $1+1+\cdots+1$ with n ones.

Let $\mathsf{Env}(\mathbb{N})$ be the set of finite partial functions $\nu : \mathsf{Var} \rightharpoonup \mathbb{N}$. Given $\varphi \in \Phi$ and $\nu \in \mathsf{Env}(\mathbb{N})$, let $\varphi[\nu]$ denote the substitution of every free variable x of φ with the number $\nu(x)$ whenever $x \in \mathrm{dom}(\nu)$. Given $\nu \in \mathsf{Env}(\mathbb{N})$ and $x \in \mathsf{Var}$ and $n \in \mathbb{N}$, let $\nu[n/x]$ denote the function $\nu' \in \mathsf{Env}(\mathbb{N})$ that agrees with ν everywhere except at x, where $\nu'(x) = n$. Let $\nu_0 \in \mathsf{Env}(\mathbb{N})$ denote the empty partial function. Note that $\varphi[\nu_0] = \varphi$ and $\varphi[\nu[n/x]] = \varphi[n/x][\nu]$ and that every $\nu \in \mathsf{Env}(\mathbb{N})$ can be written as $\nu = \nu_0[n_1/x_1]\cdots[n_k/x_k]$, so $\varphi[\nu]$ is well defined and $\varphi[\nu] \in \Phi$.

▶ **Definition 4.5.** A Borel coalgebra (S, Δ) over Φ is **uniform** if $S = Q \times \mathsf{Env}(\mathbb{N})$ for some finite set Q of *substates*, and there exists a function

$$\Delta' : Q \to (\{\mathsf{all}, \mathsf{some}\} \times Q \times \mathsf{Var}) \;\cup\; (\Phi \times Q^2) \;\cup\; \{\mathsf{accept}, \mathsf{reject}\}$$

such that for all $q \in Q$ and $\nu \in \mathsf{Env}(\mathbb{N})$,

$$\begin{array}{lll}
\Delta'(q) = (\mathsf{all}, q_1, x) & \text{implies} & \Delta(q,\nu) = (\mathsf{all}, \{(q_1, \nu[n/x]) : n \in \mathbb{N}\}),\\
\Delta'(q) = (\mathsf{some}, q_1, x) & \text{implies} & \Delta(q,\nu) = (\mathsf{some}, \{(q_1, \nu[n/x]) : n \in \mathbb{N}\}),\\
\Delta'(q) = (\varphi, q_1, q_2) & \text{implies} & \Delta(q,\nu) = (\varphi, (q_1,\nu), (q_2,\nu)),\\
\Delta'(q) = \mathsf{accept} & \text{implies} & \Delta(q,\nu) = (\mathsf{all}, \varnothing),\\
\Delta'(q) = \mathsf{reject} & \text{implies} & \Delta(q,\nu) = (\mathsf{some}, \varnothing).
\end{array}$$

Therefore Δ' completely determines the transition structure of (S, Δ) and because Q is finite, this Δ' has a finite description. A Borel automaton (S, Δ, s_0) is uniform if (S, Δ) is uniform and $s_0 = (q_0, \nu_0)$ for some $q_0 \in Q$.

Uniform Borel automata are equivalent to programs of the IND programming language introduced in [6] interpreted over a countable structure. An IND program consists of finite sequences of labeled statements of three forms:

- assignment: $\ell :\ x := \exists$ $\quad \ell :\ y := \forall$
- conditional: $\ell :\ \mathsf{if}\ \varphi\ \mathsf{then\ goto}\ \ell'$
- halt: $\ell :\ \mathsf{accept}$ $\quad \ell :\ \mathsf{reject}.$

The assignment statements represent universal and existential nondeterministic branching (simulated by $\Delta'(q) = (\mathsf{all}, q_1, x)$ and $\Delta'(q) = (\mathsf{some}, q_1, x)$). The conditional statements test a basic predicate and branch (simulated by $\Delta'(q) = (\varphi, q_1, q_2)$). The halt statements just accept and reject (simulated by $\Delta'(q) = \mathsf{accept}$ and $\Delta'(q) = \mathsf{reject}$). In light of this equivalence, we state the analogs of Theorems 3.1 and 3.2 for uniform Borel automata.

▶ **Theorem 4.6** ([6]). *Uniform Borel automata over $\mathbb{N}$ accept the inductively definable relations and decide the hyperelementary relations.*

5 Non-Wellfounded Logic

We have argued that IND programs and Borel automata provide a useful dynamic view of inductive definability and descriptive set theory. This suggests a more general non-wellfounded approach to mathematical logic, allowing non-wellfounded syntax in addition to the usual wellfounded syntax. In the previous sections, the syntax is given by *Borel coterms*, which are elements of the final Borel coalgebra. The final coalgebra is known to exist by

very general considerations [1, 2, 19], viewing oblivious Borel coalgebras as coalgebras for the set functor

$$\wp_{\omega_1}(S) + \wp_{\omega_1}(S) + S + \Phi. \tag{6}$$

We give a concrete construction in Appendix B.

In general, terms and formulas are elements of a final coalgebra whose interpretations over an algebra are computed recursively, accommodating non-halting with the use of the third logical value $\perp$. The usual wellfounded constructs are a special case.

A general framework for this approach already exists in the form of a recently developed categorical treatment of recursive function schemes [3, 12, 13]. In this treatment, one interprets coterms in an *Elgot algebra*, an algebraic structure augmented with a distinguished map $(\cdot)^\dagger$ that, applied to a system of recursive equations, produces a canonical solution to that system. Formally, a system of equations is modeled as a coalgebra $\delta : X \to A + FX$, where X is a set of indeterminates, A is an Elgot algebra over which the variables are to be interpreted, and F is a set functor describing the algebraic signature. A solution to the system is a map $\delta^\dagger : X \to A$ such that the diagram

$$\begin{array}{ccc} X & \xrightarrow{\delta^\dagger} & A \\ {\scriptstyle\delta}\downarrow & & \uparrow{\scriptstyle[\mathsf{id},\alpha]} \\ A + FX & \xrightarrow{\mathsf{id} + F\delta^\dagger} & A + FA \end{array}$$

commutes. In cases where there may be more than one solution, the desired solution is determined by $(\cdot)^\dagger$. For CPOs, this is typically the $\sqsubseteq$-least solution.

IND programs and Borel coalgebras fit this scheme with the Elgot algebra $\mathsf{3}$. For oblivious Borel coalgebras, the appropriate functor is (6). In this interpretation, the components of the coproduct (6) represent, from left to right, recursive equations of the form

$$x = \bigvee_{y \in \delta(x)} y \qquad x = \bigwedge_{y \in \delta(x)} y \qquad x = \neg y \qquad x = [\sigma \vDash \varphi].$$

A system of equations

$$\delta : X \to \mathsf{3} + \wp_{\omega_1}(X) + \wp_{\omega_1}(X) + X + \Phi$$

is essentially a Borel coalgebra with some leaf nodes annotated with $\mathsf{3}$. The canonical solution $\delta^\dagger$ is the $\sqsubseteq$-least labeling L as computed in §2.2.

It follows from general results in [3, 12, 13] that Elgot algebras of CPOs admit canonical solutions that are least in the distinguished order $\sqsubseteq$ on the domain. We can apply this theorem to obtain canonical interpretations of F-coterms over any F-algebra $\mathfrak{A}_\perp = (A_\perp, \alpha_\perp)$ obtained from an F-algebra $\mathfrak{A} = (A, \alpha)$ by augmenting with $\perp$. We show how to do this in a uniform way such that the resulting $\mathfrak{A}_\perp$ is an Elgot algebra and $\mathsf{3} = \mathsf{2}_\perp$. This is a slightly subtle construction in that the structure maps $\alpha_\perp$ are not necessarily strict, thus not morphisms in the category of CPOs in general.

Let F be a functor on Set. We can also include powerset functors in F. For example, the functor (6) is such a functor. Let (A, α) be an F-algebra. Let $A_\perp$ be the disjoint union $A \cup \{\perp\}$ ordered as a flat domain with $a \sqsubseteq b$ if either $a = b$ or $a = \perp$. Every nonempty subset $X \subseteq A$ has a meet $\bigsqcap X$.

Assume that the ordering $\sqsubseteq$ on $A_\perp$ has a natural extension to $FA_\perp$ such that F is monotone on functions $h : X \to A_\perp$ ordered pointwise. For example, we can order products

and coproducts componentwise, and for powersets, $C \sqsubseteq D$ if either (i) $C = D$, or (ii) $\bot \in C$ and $D = (C - \{\bot\}) \cup \{a\}$ for some $a \in A$.

For $x \in FA_\bot$, define

$$\alpha_\bot(x) = \prod_{\substack{x \sqsubseteq y \\ y \in FA}} \alpha(y) = \begin{cases} b, & \{\alpha(y) \mid x \sqsubseteq y \text{ and } y \in FA\} = \{b\}, \\ \bot, & |\{\alpha(y) \mid x \sqsubseteq y \text{ and } y \in FA\}| \geq 2. \end{cases}$$

In other words, thinking of $\bot$ as "don't know", we replace $\bot$ with elements of A in all possible ways and compute α on the resulting inputs; if there is only one result, i.e., there is no ambiguity, then that is the result of $\alpha_\bot$, otherwise it is still $\bot$.

▶ **Example 5.1.** $3 = 2_\bot$.

▶ **Lemma 5.2.** *$\alpha_\bot : FA_\bot \to A_\bot$ is monotone.*

▶ **Theorem 5.3.** *For any $\mathfrak{A}$, $\mathfrak{A}_\bot$ is an Elgot algebra.*

Proof. The least solution is constructed inductively by a straightforward generalization of the inductive procedure in §2.2. ◀

An F-coterm over variables X is an element of the final coalgebra $\gamma : CX \to X + FCX$. For F the functor (6) and $X = \varnothing$, this is the final coalgebra C constructed in Appendix B. It follows from results of [3, 12, 13] that any F-coterm in CX satisfies a universality property akin to free algebras in the sense that any set map $\theta : X \to A_\bot$ extends to a least F-algebra morphism $h : CX \to A_\bot$ such that the following diagram commutes:

$$\begin{array}{ccc} CX & \xrightarrow{\quad h \quad} & A_\bot \\ {\scriptstyle\gamma}\downarrow & & \uparrow{\scriptstyle[\mathsf{id}, \alpha]} \\ X + FCX & \xrightarrow{\theta + Fh} & A_\bot + FA_\bot \end{array}$$

For F the functor (6), $X = \varnothing$, and $A_\bot = 3$, the map h coincides with the least labeling L_σ computed in §2.2.

6 Conclusion and Future Work

We have introduced *Borel coalgebras* and *Borel automata* as a computational approach to basic descriptive set theory, recasting some basic results of that theory in a more computational framework. We have also shown how these ideas lead to a general notion of non-wellfounded logic in which syntactic objects such as terms and formulas are elements of a final coalgebra.

For the future, we would like to relate these results to the Borel machines of Klarlund [8, 9], which are more conventional automata on infinite words over a finite alphabet with Borel acceptance conditions on the words. We would also like to explore the use of computational methods in other results of descriptive set theory such as the determinacy of games. Finally, we would like to investigate deductive systems, completeness theorems, and decision procedures for suitable fragments of non-wellfounded logic.

References

1. P. Aczel and P.F. Mendler. A final coalgebra theorem. In *Category Theory and Computer Science*, volume 389 of *Lecture Notes in Computer Science*, pages 357–365. Springer, 1989.
2. J. Adámek. On final coalgebras of continuous functors. *Theor. Comput. Sci.*, 294:3–29, 2003.
3. Jiří Adámek, Stefan Milius, and Jiří Velebil. Elgot algebras. *Log. Methods Comput. Sci.*, 2(5:4):1–31, 2006.
4. J. Barwise, R. Gandy, and Y. Moschovakis. The next admissible set. *J. Symbolic Logic*, 36:108–120, 1971.
5. Jon Barwise. *Admissible Sets and Structures*. North-Holland, 1975.
6. David Harel and Dexter Kozen. A programming language for the inductive sets, and applications. *Information and Control*, 63(1–2):118–139, 1984.
7. H. Jerome Keisler. *Descriptive Set Theory*, volume 62 of *Studies in Logic and the Foundations of Mathematics*. North-Holland, 1971.
8. Nils Klarlund. *Progress Measures and Finite Arguments for Infinite Computations*. PhD thesis, Cornell University, August 1990.
9. Nils Klarlund. The limit view of infinite computations. In Bengt Jonsson and Joachim Parrow, editors, *5th Int. Conf. Concurrency Theory (CONCUR'94)*, volume 836 of *Lecture Notes in Computer Science*, pages 351–366, Uppsala, Sweden, August 1994. Springer.
10. Dexter Kozen. Computational inductive definability. *Annals of Pure and Applied Logic*, 126(1–3):139–148, April 2004. Special issue: *Provinces of logic determined. Essays in the memory of Alfred Tarski*. Zofia Adamowicz, Sergei Artemov, Damian Niwinski, Ewa Orlowska, Anna Romanowska, and Jan Wolenski (eds.).
11. D. A. Martin. Descriptive set theory. In J. Barwise, editor, *Handbook of Mathematical Logic*, volume 90 of *Studies in Logic and the Foundations of Mathematics*, pages 783–815. Elsevier, 1977.
12. Stefan Milius and Lawrence S. Moss. The category theoretic solution of recursive program schemes. *Theoret. Comput. Sci.*, 366:3–59, 2006.
13. Stefan Milius and Lawrence S. Moss. Corrigendum to: "the category theoretic solution of recursive program schemes". *Theoret. Comput. Sci.*, 403(2–3):409–415, 2008.
14. Y. N. Moschovakis. Abstract first order computability I. *Trans. Amer. Math. Soc.*, 138:427–464, 1969.
15. Y. N. Moschovakis. *Elementary Induction on Abstract Structures*. North-Holland, 1974.
16. Yiannis N. Moschovakis. The suslin-kleene theorem for countable structures. *Duke Math. J.*, 37(2):341–352, 06 1970.
    `http://dx.doi.org/10.1215/S0012-7094-70-03744-0`
17. Yiannis N. Moschovakis. *Descriptive Set Theory*, volume 100 of *Studies in Logic and the Foundations of Mathematics*. North-Holland, 1980.
18. Lawrence S. Moss. Coalgebraic logic. *Annals of Pure and Applied Logic*, 96(1–3):277–317, 1999. Festschrift on the occasion of Professor Rohit Parikh's 60th birthday. Errata: *Ibid.*, 99(1–3):241–259, 1999.
19. Luigi Santocanale. Logical construction of final coalgebras. In H. Peter Gumm, editor, *Proc. Workshop on Coalgebraic Methods in Computer Science, Warsaw, Poland*, volume 82 of *Electronic Notes in Theoretical Computer Science*, pages 1–20. Elsevier, April 2003.

A Omitted Proofs

Proof of Theorem 3.1. Most of the proof is given in the main text, but we omitted the argument that the acceptance condition (1) can be transformed to Π^1_1 form (3). Here is the full argument.

Without loss of generality assume S is oblivious. The acceptance condition (1) is

$$\forall L \ \ (\tau_\sigma(L) \sqsubseteq L \Rightarrow L(s_0) = 1)\,, \tag{7}$$

where

$$\tau_\sigma(L)(s) = \begin{cases} \bigvee_{t\in\delta(s)} L(t), & \ell(s) = \mathsf{some}, \\ \bigwedge_{t\in\delta(s)} L(t), & \ell(s) = \mathsf{all}, \\ \neg L(\delta(s)), & \ell(s) = \mathsf{not}, \\ [\mathfrak{A}, \sigma \vDash \varphi], & \ell(s) = \varphi. \end{cases} \tag{8}$$

where $[\mathfrak{A}, \sigma \vDash \varphi] \in 2$ denotes the truth value of this basic predicate. Because the set of states reachable from the start state s_0 is countable and duplication of siblings does not affect acceptance or rejection, we can assume without loss of generality that $S = \omega^*$, $s_0 = \varepsilon$, and $\delta(s) = \{sn \mid n \in \omega\}$. Then (8) becomes

$$\tau_\sigma(L)(s) = \begin{cases} \bigvee_{n\in\omega} L(sn), & \ell(s) = \mathsf{some}, \\ \bigwedge_{n\in\omega} L(sn), & \ell(s) = \mathsf{all}, \\ \neg L(s0), & \ell(s) = \mathsf{not}, \\ [\sigma \vDash \varphi], & \ell(s) = \varphi. \end{cases}$$

The prefixpoint condition $\tau_\sigma(L) \sqsubseteq L$ of (7) expands to

$$\forall s \left(\begin{array}{ll} & (\ell(s) = \mathsf{some} \Rightarrow \bigvee_{n\in\omega} L(sn) \sqsubseteq L(s)) \\ \wedge & (\ell(s) = \mathsf{all} \Rightarrow \bigwedge_{n\in\omega} L(sn) \sqsubseteq L(s)) \\ \wedge & (\ell(s) = \mathsf{not} \Rightarrow \neg L(s0) \sqsubseteq L(s)) \\ \wedge & (\ell(s) = \varphi \Rightarrow [\sigma \vDash \varphi] \sqsubseteq L(s)) \end{array} \right)$$

where s ranges over ω^*. This is equivalent to

$$\forall s \left(\begin{array}{ll} & (\ell(s) = \mathsf{some} \wedge (\forall n\ L(sn) = 0) \Rightarrow L(s) = 0) \\ \wedge & (\ell(s) = \mathsf{some} \wedge (\exists n\ L(sn) = 1) \Rightarrow L(s) = 1) \\ \wedge & (\ell(s) = \mathsf{all} \wedge (\forall n\ L(sn) = 1) \Rightarrow L(s) = 1) \\ \wedge & (\ell(s) = \mathsf{all} \wedge (\exists n\ L(sn) = 0) \Rightarrow L(s) = 0) \\ \wedge & (\ell(s) = \mathsf{not} \wedge L(s0) \neq \bot \Rightarrow L(s) = \neg L(s0)) \\ \wedge & (\ell(s) = \varphi \Rightarrow L(s) = [\sigma \vDash \varphi]) \end{array} \right).$$

Writing this in prenex form,

$$\forall s\ \exists k\ \exists m\ \forall n \left(\begin{array}{ll} & (\ell(s) = \mathsf{some} \wedge L(sk) = 0 \Rightarrow L(s) = 0) \\ \wedge & (\ell(s) = \mathsf{some} \wedge L(sn) = 1 \Rightarrow L(s) = 1) \\ \wedge & (\ell(s) = \mathsf{all} \wedge L(sm) = 1 \Rightarrow L(s) = 1) \\ \wedge & (\ell(s) = \mathsf{all} \wedge L(sn) = 0 \Rightarrow L(s) = 0) \\ \wedge & (\ell(s) = \mathsf{not} \wedge L(s0) \neq \bot \Rightarrow L(s) = \neg L(s0)) \\ \wedge & (\ell(s) = \varphi \Rightarrow L(s) = [\sigma \vDash \varphi]) \end{array} \right)$$

which we abbreviate to

$$\forall s\ \exists k\ \exists m\ \forall n\ \psi(\sigma, L, s, k, m, n).$$

Skolemizing, we can write this as

$$\exists f\ \exists g\ \forall s\ \forall n\ \psi(\sigma, L, s, f(s), g(s), n)$$

where $f, g : \omega^* \to \omega$. The acceptance condition (7) can now be written

$$\begin{aligned}&\forall L\ ((\exists f\ \exists g\ \forall s\ \forall n\ \psi(\sigma, L, s, f(s), g(s), n)) \Rightarrow L(\varepsilon) = 1)\\ &\Leftrightarrow\ \forall L\ \forall f\ \forall g\ \exists s\ \exists n\ (\psi(\sigma, L, s, f(s), g(s), n) \Rightarrow L(\varepsilon) = 1).\end{aligned}$$

This formula is in the desired Π^1_1 form (3). ◀

Proof of Theorem 4.3. ($\Rightarrow$) Let $L(S, b, n) = \{a \in \mathfrak{A} : \tau_\sigma^{(n)}(L_0)(s_0) = b \text{ where } \sigma(x) = a\}$. Note that $L(S, b, n)$ is clopen for all $b \in \{0, 1\}$ and $n < \omega$, by induction on n. Therefore $L(S)$ is open:

$$\begin{aligned}L(S) &= \{a \in \mathfrak{A} : S \text{ accepts } a\}\\ &= \{a \in \mathfrak{A} : L_\sigma(s_0) = 1 \text{ where } \sigma(x) = a\}\\ &= \{a \in \mathfrak{A} : \textstyle\bigsqcup_{n<\omega} \tau_\sigma^{(n)}(L_0)(s_0) = 1 \text{ where } \sigma(x) = a\} \qquad \text{(by Lemma 4.2)}\\ &= \textstyle\bigcup_{n<\omega} L(S, 1, n).\end{aligned}$$

($\Leftarrow$) Let $A \subseteq \mathfrak{A}$ be open and let $B_1, B_2, B_3 \ldots$ be basic clopen sets such that $A = \bigcup_{n<\omega} B_n$. Then A is accepted by the finitary Borel automaton $(\mathbb{N} \cup \{s_{\mathsf{accept}}\}, \Delta, 0)$ defined by:

$$\begin{aligned}&n : \mathsf{if}\ x \in B_n\ \mathsf{then\ accept\ else}\ (n+1)\\ &\mathsf{accept} : \mathsf{all}\ \varnothing\end{aligned}$$

One can show that $L_\sigma(n) = 1$ if and only if $\sigma(x) \in \bigcup_{i=n}^{\omega} B_i$ and therefore $L(S) = A$. ◀

Proof of Theorem 4.4. This follows from a compactness argument. Let $L(S, b, n) = \{a \in \mathfrak{A} : \tau_\sigma^{(n)}(L_0)(s_0) = b \text{ where } \sigma(x) = a\}$. Since S halts on all inputs, and by Lemma 4.2, we know that $L(S) = \bigcup_{n\in\omega} L(S, 1, n)$ and $(\mathfrak{A} \setminus L(S)) = \bigcup_{n\in\omega} L(S, 0, n)$. These are both open, so they are both closed, and so they are both compact subsets of $\mathfrak{A}$. By compactness and the fact that $L(S, b, n) \subseteq L(S, b, n+1)$, there is some n such that $L(S) = L(S, 1, n)$ and $(\mathfrak{A} \setminus L(S)) = L(S, 0, n)$. Thus S halts in n steps for all inputs. ◀

B Construction of the Final Borel Coalgebra

In this section we construct the final Borel coalgebra. The final coalgebra is known to exist by very general considerations [1, 2, 19], viewing oblivious Borel coalgebras as coalgebras for the set functor

$$\wp_{\omega_1}(S) + \wp_{\omega_1}(S) + S + \Phi$$

but we give a concrete construction, because we wish to use it as an abstract syntax for non-wellfounded logic. We work with oblivious Borel coalgebras (§2.4) because this simplifies the development.

▶ **Definition 2.1.** A *Borel coterm* is an element of the final Borel coalgebra.

Coterms can be constructed explicitly as labeled trees modulo the maximum autobisimulation. A *term* is a well founded coterm. Terms need not be finite. A *regular coterm* is one with a finite representation; that is, one represented by an element of a finite Borel coalgebra under the canonical morphism. Regular coterms need not be well founded.

B.1 Labeled Trees

A particular Borel coalgebra is the set of *labeled trees*. This is not quite the final coalgebra, as we must still identify bisimilar trees; but it is close to it and a good syntactic representation. A *labeled tree* is a partial function $t : \omega^* \rightharpoonup \{\mathsf{all}, \mathsf{some}, \mathsf{not}\} \cup \Phi$ such that

- $\operatorname{dom} t$ is nonempty and prefix-closed (that is, if $x \leq y \in \operatorname{dom} t$, then $x \in \operatorname{dom} t$);
- if $x \in \operatorname{dom} t$ and $t(x) = \mathsf{not}$, then $xn \in \operatorname{dom} t$ for exactly one $n \in \omega$; and
- if $x \in \operatorname{dom} t$ and $t(x) \in \Phi$, then $xn \notin \operatorname{dom} t$ for all $n \in \omega$.

Any such t must be defined on the empty string ε at least, which is the root of the tree.

If $x \in \operatorname{dom} t$, the *subtree of t rooted at x* is the labeled tree $t{\restriction}x = \lambda y\,.\,t(xy)$. Note that if $x \leq y$ and $y \in \operatorname{dom} t$, then $t{\restriction}y$ is a subtree of $t{\restriction}x$. The set of labeled trees forms a Borel coalgebra as follows:

If $t(\epsilon)$ is...	then...
all	$t : \mathsf{all}\ \{t{\restriction}n : n \in \omega \cap \operatorname{dom} t\}$
some	$t : \mathsf{some}\ \{t{\restriction}n : n \in \omega \cap \operatorname{dom} t\}$
not	$t : \mathsf{not}\ t{\restriction}n \quad \text{where} \quad n \in \omega \cap \operatorname{dom} t$
φ	$t : \varphi$

Equivalently,

$$\ell(t) = t(\epsilon) \qquad \text{and} \qquad \delta(t) = \{t{\restriction}n : n \in \omega \cap \operatorname{dom} t\}.$$

B.2 Bisimulation

A *bisimulation* between two oblivious Borel coalgebras (S, Δ) and (T, Δ) is a binary relation $R \subseteq S \times T$ such that for all $s \in S$ and $t \in T$, if $R(s, t)$, then

1. $\ell(s) = \ell(t)$
2. for all $s' \in \delta(s)$, there exists $t' \in \delta(t)$ such that $R(s', t')$
3. for all $t' \in \delta(t)$, there exists $s' \in \delta(s)$ such that $R(s', t')$

We say that two states are *bisimilar* if there is a bisimulation relating them. A bisimulation between a Borel coalgebra and itself is called an *autobisimulation*. The identity relation is an autobisimulation, and the union of any set of bisimulations is a bisimulation. Thus there is a unique coarsest autobisimulation $\equiv$ on any coalgebra, which is the union of all autobisimulations. Two states of the same coalgebra are bisimilar iff they are related by $\equiv$. It is an equivalence relation with classes $[s] = \{t \mid s \equiv t\}$. The quotient $S/{\equiv}$ with states $[s]$ is again a Borel coalgebra with $\ell([s]) = \ell(s)$ and $\delta([s]) = \{[t] \mid t \in \delta(s)\}$. The properties of autobisimulation ensure that ℓ and δ are well defined on $S/{\equiv}$. Moreover, the map $s \mapsto [s]$ is a morphism of Borel coalgebras, and s and $[s]$ are bisimilar.

B.3 Construction of the Canonical Morphism

▶ **Theorem 2.2.** *The coalgebra of labeled trees modulo bisimilarity is the final Borel coalgebra.*

Proof of Theorem 2.2. We construct the canonical map from an arbitrary oblivious Borel coalgebra (S, ℓ, δ) to the final coalgebra.

For $s \in S$, let $\mathsf{desc}\, s$ be the set of *descendants* of s, the smallest set containing s and closed under δ. The set $\mathsf{desc}\, s$ is countable, so let us assume without loss of generality that $\mathsf{desc}\, s \subseteq \omega$. A *finite run from s* is a sequence $n_0 \cdots n_k$, $k \geq 0$, such that $s = n_0$ and

$n_i \in \delta(n_{i-1})$, $1 \leq i \leq k$. Let t be the tree whose domain is the set of all $n_1 \cdots n_k \in \omega^*$ such that $n_0 \cdots n_k$ is a finite run from s with $t(n_1 \cdots n_k) = \ell(n_k)$ and $t(\varepsilon) = \ell(n_0)$. Every oblivious coalgebra generates a set of runs of this form, which is identical to one of the labeled trees constructed in §B.1. The canonical map maps s to the image of this subtree in the bisimilarity quotient. ◀