Logical Methods in Computer Science
Vol. 3 (3:1) 2007, pp. 1–23
www.lmcs-online.org

Submitted Nov. 24, 2006
Published Jul. 4, 2007

A FINITE SEMANTICS OF SIMPLY-TYPED LAMBDA TERMS FOR INFINITE RUNS OF AUTOMATA

KLAUS AEHLIG

Department of Computer Science, University of Wales Swansea, Swansea SA2 8PP, United Kingdom
e-mail address: k.t.aehlig@swan.ac.uk

Abstract. Model checking properties are often described by means of finite automata. Any particular such automaton divides the set of infinite trees into finitely many classes, according to which state has an infinite run. Building the full type hierarchy upon this interpretation of the base type gives a finite semantics for simply-typed lambda-trees.

A calculus based on this semantics is proven sound and complete. In particular, for regular infinite lambda-trees it is decidable whether a given automaton has a run or not. As regular lambda-trees are precisely recursion schemes, this decidability result holds for arbitrary recursion schemes of arbitrary level, without any syntactical restriction.

1. Introduction and Related Work

The lambda calculus [5] has long been used as model of computation. In its untyped form it is Turing complete. Even though models of the untyped lambda calculus are known, restricting it to a typing discipline allows for more specific models. The simply-typed lambda calculus has a straight forward set-theoretic semantics.

Quite early on, not only finite but also infinite lambda-terms have been considered. For example, Barendregt [5] introduced the concept of “Böhm trees” as a generalised concept of normal forms for lambda-terms where normalisation does not necessarily terminate, but still might produce a growing normal prefix; for example the term $Y(\lambda zx.xz)$ has the Böhm tree $\lambda x.x(\lambda x.x(\lambda x.x\ldots))$.

Since Rabin [16] showed the decidability of the monadic second order (MSO) theory of the infinite binary tree this result has been applied and extended to various mathematical structures, including algebraic trees [8] and a hierarchy of graphs [7] obtained by iterated unfolding and inverse rational mappings from finite graphs. The interest in these kind of structures arose in recent years in the context of verification of infinite state systems [13, 18].

2000 ACM Subject Classification: F.3.2.
Key words and phrases: Recursion Schemes, infinitary lambda calculus, automata.
Partially supported by grant EP/D03809X/1 of the British Engineering and Physical Sciences Research Council (EPSRC). Part of this article was written while Klaus Aehlig was affiliated with the University of Toronto and supported by grant Ae 102-1/1 of the “Deutsche Forschungsgemeinschaft” (DFG).

Recently Knapik, Niwiński and Urzyczyn [10] showed that the monadic second order theory of any infinite tree generated by a level-2 grammar satisfying a certain "safety" condition is decidable. Later they generalised [11] this result to grammars of arbitrary levels, but still requiring the "safety" condition. In particular, the question was left open whether a "safety" constraint is necessary to obtain decidability. In this article we will give a partial answer.

It should be noted that trees given by higher-order grammars can also be understood as trees given by simply-typed infinite, but regular, lambda terms. The "safety" condition guarantees that beta-reduction can be carried out in such a way that variables never have to be renamed in the process of substitution. This obviously is a property related to operational aspects of computation. Our approach to avoid the need for such a restriction is therefore to search for a *denotational* semantics. Denotational approaches tend to be less vulnerable to the need of requiring specific *operational* properties.

To obtain effective constructions, like an effective semantics, it is useful to have a concrete representation of the properties to be verified. Finite automata are a standard tool to do so. In this article we concentrate on automata with trivial acceptance condition. These automata do not exhaust the full of MSO but, as we shall see, are able to express a reasonable set of safety properties.

Their advantage, however, is that they seem particularly suited for a denotational approach. The reason is, that the "interface" is particularly simple. In order to combine two partial runs into a longer run, the only thing we have to look at is the state in which the automaton arrives.

Based on this intuition we construct a semantics for the simple types. Actually, we use the standard set-theoretic semantics. Hence the only thing we have to specify is the interpretation of the base type. Following the discussion above, we describe a term of base type by the set of states a given automaton can start a run on the tree denoted by that term.

More precisely, we consider the following problem.

> Given a, possibly infinite, simply-typed lambda-tree t of base type, and given a non-deterministic tree automaton $\mathfrak{A}$. Does $\mathfrak{A}$ have a run on the normal form of t?

The idea is to provide a "proof" of a run of $\mathfrak{A}$ on the normal form of t by annotating each subterm of t with a semantical value describing how this subterm "looks, as seen by $\mathfrak{A}$". Since, in the end, all the annotations come from a fixed finite set, the existence of such a proof is decidable.

The idea of a "proof" that a given automaton has a run on a tree is used, at least implicitly, in the work by Aehlig, de Miranda and Ong [4]. This work also gives an affirmative answer to the question of the decidability for the full MSO theory for trees generated by level-two recursion schemes.

Very recently, simultaneously and independently, Luke Ong could give an affirmative answer [15] for trees generated by recursion schemes of arbitrary level, still deciding the full MSO theory; he thus obtained a stronger result in what concerns decidability. His result is based on game semantics [9] and is technically quite involved. Therefore the author believes that his conceptually more simple approach still is of worth. Moreover, the novel finitary semantics for the simple types introduced in this article, and the sound and complete proof system to show the existence of a run of an automaton seem to be of independent interest. An extended abstract [1] of this article appeared in the proceedings of CSL '06.

This article is organised as follows. In Section 2 we formally introduce automata with trivial acceptance condition and study their languages. We also prove the closure of these languages under the modality "globally". We also show that properties based on the modality "eventually" are not expressible. In Section 3 we introduce infinitary simply-typed lambda trees and in Section 4 we introduce recursion schemes as a means to describe regular lambda trees. This also shows that some lambda trees have a representation that is not only effective, but also quite natural. In Section 5 we explain continuous normalisation for the lambda calculus. The use of continuous normalisation is twofold. On the one hand, it allows simpler definitions and proofs, as one layer of input corresponds precisely to one layer of output. On the other hand, it is simply a necessity in order to have a well-defined normal form in the presence of non-terminating computations due to the infinitary nature of our lambda trees. Section 6 introduces the finitary semantics and the proof system; Sections 7 and 8 are devoted to the proofs of its soundness and completeness. Finally, in Section 9, we put the results together to obtain the mentioned decidability result.

2. Automata with Trivial Acceptance Condition

We assume a set of *letters* or *terminals* be given to us as a primitive notion. We use $\mathfrak{f}$ to range over letters. Each letter $\mathfrak{f}$ is associated an *arity* $\sharp(\mathfrak{f}) \in \mathbb{N}$.

Definition 2.1. For Σ a set of terminals, a Σ-*term* is a, not necessarily well-founded, tree labelled with elements of Σ where every node labelled with $\mathfrak{f}$ has $\sharp(\mathfrak{f})$ many children.

A Σ-*language* is any subset of the set of all Σ-terms. We use the term *language* if Σ is understood.

Example 2.2. Let $\Sigma' = \{\mathtt{f}, \mathtt{g}, \mathtt{a}\}$ with $\mathtt{f}$, $\mathtt{g}$ and $\mathtt{a}$ of arities 2, 1, and 0, respectively. Figure 1 shows two Σ'-terms.

Definition 2.3 (Trivial Automata)**.** A non-deterministic tree automaton with trivial acceptance condition over the alphabet Σ, or a "trivial automaton" for short, is given by

- a finite set Q of "states",
- a set $I \subset Q$ of "initial states", and
- a transition function $\delta\colon Q \times \Sigma \to \mathfrak{P}((Q \cup \{*\})^N)$.

Here $N = \max\{\sharp(\mathfrak{g}) \mid \mathfrak{g} \in \Sigma\}$ is the maximal arity and we require $\delta(q, \mathfrak{g}) \subset Q^{\sharp(\mathfrak{g})} \times \{*\}^{N-\sharp(\mathfrak{g})}$ whenever $q \in Q$ and $\mathfrak{g} \in \Sigma$.

Definition 2.4 (Run of a Trivial Automaton)**.** If t is Σ-term, and $\mathfrak{A}$ a trivial automaton over Σ, then a *run* (also "an infinite run") *of* $\mathfrak{A}$ *on* t *starting in state* q is a mapping r from the nodes of t to Q, such that the root is mapped to q, and, whenever p is a $\mathfrak{f}$-labelled node in t and $p_1, \ldots, p_{\sharp(\mathfrak{f})}$ are the children of p, then $(r(p_1), \ldots, r(p_{\sharp(\mathfrak{f})}), *, \ldots, *) \in \delta(r(p), \mathfrak{f})$.

A *run up to level* n *starting in state* q is a mapping from all nodes of t with distance at most n to Q such that the above condition holds for all nodes $p, \overrightarrow{p}$ in the domain of r, i.e., whenever a node p is $\mathfrak{f}$-labelled and its children $p_1, \ldots, p_{\sharp(\mathfrak{f})}$ have distance at most n to the root, then $(r(p_1), \ldots, r(p_{\sharp(\mathfrak{f})}), *, \ldots, *) \in \delta(r(p), \mathfrak{f})$.

A *run* or a *run up to level* n, is a run or a run up to level n starting in some initial state.

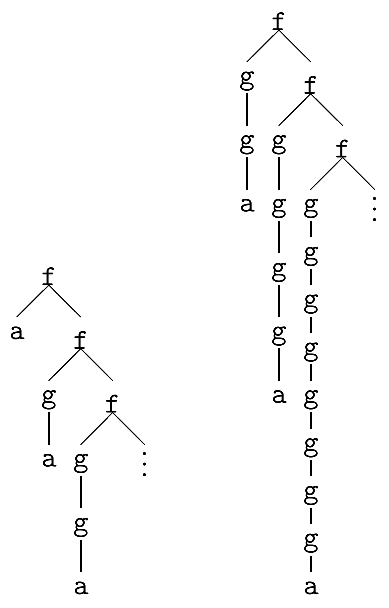


Figure 1: Two $\{\mathtt{f},\mathtt{g},\mathtt{a}\}$-terms.

We write $\mathfrak{A}, q \models^n t$ to denote that $\mathfrak{A}$ has a run on t up to level n starting in state q. We write $\mathfrak{A}, q \models^\infty t$ to denote that $\mathfrak{A}$ has a run on t starting in state q. We write $\mathfrak{A} \models^n t$ to denote that $\mathfrak{A}$ has a run up to level n on t and we write $\mathfrak{A} \models^\infty t$ to denote that $\mathfrak{A}$ has a run on t.

Remark 2.5. Trivially, every automaton has a run up to level 0 on every term starting in every state. Also immediate from the definition we see that, if $\mathfrak{A}$ has a run up to level n on t and $m \leq n$ then $\mathfrak{A}$ has a run up to level m on t.

Remark 2.6. By König's Lemma $\mathfrak{A}$ has a run on t if and only if $\mathfrak{A}$ has a run up to level n on t for every $n \in \mathbb{N}$.

Example 2.7. Continuing Example 2.2 consider the property

"Every maximal chain of letters $\mathtt{g}$ has even length".

It can be expressed by an automaton with two states $Q = \{q_2, q_1\}$ where q_2 means that an even number of $\mathtt{g}$s has been passed on the path so far, and q_1 means that the maximal chain of $\mathtt{g}$s passed has odd length. Then the initial state is q_2 and the transition function is as follows.

$$\begin{array}{rclcrcl}
\delta(\mathtt{f}, q_2) & = & \{(q_2, q_2)\} & \qquad & \delta(\mathtt{f}, q_1) & = & \emptyset \\
\delta(\mathtt{g}, q_2) & = & \{(q_1, *)\} & & \delta(\mathtt{g}, q_1) & = & \{(q_2, *)\} \\
\delta(\mathtt{a}, q_2) & = & \{(*, *)\} & & \delta(\mathtt{a}, q_1) & = & \emptyset
\end{array}$$

Note that this automaton has an infinite run on the second tree in Figure 1, whereas it has a run only up to level 3 on the first one.

Definition 2.8 ($\mathcal{L}(\mathfrak{A})$)**.** If $\mathfrak{A}$ is a trivial automaton over the alphabet Σ then by $\mathcal{L}(\mathfrak{A})$ we denote the language of $\mathfrak{A}$, that is, the set

$$\mathcal{L}(\mathfrak{A}) = \{t \mid \mathfrak{A} \models^\infty t\}$$

of all terms t such that $\mathfrak{A}$ has a run on t.

Proposition 2.9. *There exists a trivial automaton that accepts a tree if and only if its root is labelled by the terminal* $\mathfrak{f}$.

Proof. Let q_1 be an all-accepting state, i.e., $\delta(q_1, \mathfrak{g}) = \{(q_1, \dots, q_1, *, \dots, *)\}$ for all $\mathfrak{g} \in \Sigma$. Let q_0 be the only initial state, and set $\delta(q_0, \mathfrak{f}) = \{(q_1, \dots, q_1, *, \dots, *)\}$ and $\delta(q_0, \mathfrak{g}) = \emptyset$ for $\mathfrak{g} \neq \mathfrak{f}$. ☐

Lemma 2.10. *If* $\mathfrak{A}_0$ *and* $\mathfrak{A}_1$ *are trivial automata, then there is a trivial automaton* $\mathfrak{A}$ *with* $\mathcal{L}(\mathfrak{A}) = \mathcal{L}(\mathfrak{A}_0) \cup \mathcal{L}(\mathfrak{A}_1)$.

Proof. Let $\mathfrak{A}_i$ have state set Q_i, initial states I_i and transition δ_i. Assume, without loss of generality, that Q_0 and Q_1 are disjoint. Then $\mathfrak{A}$ is given by the following data. State set is $Q = Q_0 \cup Q_1$, initial states are $I = I_0 \cup I_1$ and the transition function δ is defined by $\delta(q, \mathfrak{f}) = \delta_i(q, \mathfrak{f})$ for $q \in Q_i$. ☐

Lemma 2.11. *If* $\mathfrak{A}_0$ *and* $\mathfrak{A}_1$ *are trivial automata, then there is a trivial automaton* $\mathfrak{A}$ *with* $\mathcal{L}(\mathfrak{A}) = \mathcal{L}(\mathfrak{A}_0) \cap \mathcal{L}(\mathfrak{A}_1)$.

Proof. Let $\mathfrak{A}_i$ have state set Q_i, initial states I_i and transition δ_i. Set $Q = Q_0 \times Q_1$, $I = I_0 \times I_1$ and define $\delta \colon (Q_0 \times Q_1) \times \Sigma \to \mathfrak{P}((Q_0 \times Q_1 \cup \{*\})^N)$ by $\delta((q, q'), \mathfrak{f}) = \{((q_0, q'_0), \dots, (q_{\sharp(\mathfrak{f})}, q'_{\sharp(\mathfrak{f})}), *, \dots, *) \mid (q_0, \dots, q_{\sharp(\mathfrak{f})}, \dots) \in \delta_0(q, \mathfrak{f}) \wedge (q'_0, \dots, q'_{\sharp(\mathfrak{f})}, \dots) \in \delta_1(q, \mathfrak{f})\}$. Then Q, I and δ define an automaton $\mathfrak{A}$ as desired. ☐

Non-determinism immediately provides us with closure under projection of the alphabet; we'll give a precise definition of this property.

Definition 2.12. If Σ and $\overline{\Sigma}$ are sets of terminals, a *projection from* Σ *to* $\overline{\Sigma}$, is a mapping $\pi \colon \Sigma \to \overline{\Sigma}$ such that $\sharp(\pi(\mathfrak{f})) = \sharp(\mathfrak{f})$ for all $\mathfrak{f} \in \Sigma$. If t is a Σ-term and π is a projection from Σ to $\overline{\Sigma}$, then by $\pi(t)$ we denote the $\overline{\Sigma}$-term that is obtained from t by replacing every label $\mathfrak{f}$ by $\pi(\mathfrak{f})$.

Remark 2.13. In Definition 2.12 the condition on the arity is necessary to ensure that $\pi(t)$ is a well-formed $\overline{\Sigma}$-tree, i.e., every node $\mathfrak{g}$-labelled node has $\sharp(\mathfrak{g})$ many children.

Lemma 2.14. *If* Σ *and* $\overline{\Sigma}$ *are sets of terminals,* π *is a projection from* Σ *to* $\overline{\Sigma}$*, and* $\mathfrak{A}$ *is a trivial automaton* Σ*, then there is a trivial automaton* $\mathfrak{A}_\pi$ *such that*

$$\mathcal{L}(\mathfrak{A}_\pi) = \{\pi(t) \mid t \in \mathcal{L}(\mathfrak{A})\} .$$

Proof. Let $\mathfrak{A}$ have state set Q, initial states I and transition δ. Then a possible automaton $\mathfrak{A}_\pi$ is given by the same set Q of states and the same set I of initial state, but with transition function δ_π defined by $\delta_\pi(q, \mathfrak{g}) = \bigcup\{\delta(q, \mathfrak{f}) \mid \mathfrak{f} \in \Sigma, \pi(\mathfrak{f}) = \mathfrak{g}\}$. ☐

Another obvious closure property of the languages of trivial automata are the temporal "next" operators.

Definition 2.15 ($\mathsf{EX}\mathcal{L}$, $\mathsf{AX}\mathcal{L}$)**.** If $\mathcal{L}$ is a language we define the languages

$$\mathsf{EX}\mathcal{L} = \{\mathfrak{f} t_1 \dots t_{\sharp(\mathfrak{f})} \mid \exists i . t_i \in \mathcal{L}\}$$

and

$$\mathsf{AX}\mathcal{L} = \{\mathfrak{f} t_1 \dots t_{\sharp(\mathfrak{f})} \mid \forall i . t_i \in \mathcal{L}\} .$$

Lemma 2.16. *If* $\mathfrak{A}$ *is a trivial automaton, then there exist trivial automata* $\mathfrak{A}_{\mathsf{EX}}$ *and* $\mathfrak{A}_{\mathsf{AX}}$ *with* $\mathcal{L}(\mathfrak{A}_{\mathsf{EX}}) = \mathsf{EX}\mathcal{L}(\mathfrak{A})$ *and* $\mathcal{L}(\mathfrak{A}_{\mathsf{AX}}) = \mathsf{AX}\mathcal{L}(\mathfrak{A})$.

Proof. To construct $\mathfrak{A}_{\mathsf{AX}}$, add a new state q_0 to the state set of $\mathfrak{A}$. This new state will be the only initial state of $\mathfrak{A}_{\mathsf{AX}}$. Extend the transition function δ by setting

$$\delta(q_0,\mathfrak{f}) = \{(q_1,\ldots,q_{\sharp(\mathfrak{f})},*,\ldots,*) | q_1,\ldots,q_{\sharp(\mathfrak{f})} \in I\}$$

where I is the set of initial states of $\mathfrak{A}$.

To construct $\mathfrak{A}_{\mathsf{EX}}$ from $\mathfrak{A}$ add a new state q_0, which will be the only initial state of the new automaton, and add a new all-accepting state q_f. Extend δ by setting

$$\begin{aligned}\delta(q_0,\mathfrak{f}) = \{ \; & (q_i, q_f \ldots, q_f, q_f, *, \ldots, *),\\ & (q_f, q_i \ldots, q_f, q_f, *, \ldots, *),\\ & \ldots\\ & (q_f, q_f \ldots, q_i, q_f, *, \ldots, *),\\ & (q_f, q_f \ldots, q_f, q_i, *, \ldots, *) \; | q_i \in I\}\end{aligned}$$

where I is the set of initial states of $\mathfrak{A}$. □

Definition 2.17 ($p \in t$, $t|_p$, Path)**.** We use $p \in t$ to express that p is a node in t. In this case we write $t|_p$ for the subterm of t whose root is p.

A *path* in t is a maximal set P of nodes in t such that if a node $p \in t$ different from the root is in P, then so is its parent, and such that for every node in P at most one of its children is in P.

Remark 2.18. Immediately from the definition of a path we note that if P is a path in t and $p \in P$ has a child in t then some child of p has to be in P.

Definition 2.19. If $\mathcal{L}$ is a language we define the languages

$$\mathsf{EG}\mathcal{L} = \{t \mid \exists P(P \text{ path in } t \wedge \forall p \in P. t|_p \in \mathcal{L})\}$$

and

$$\mathsf{AG}\mathcal{L} = \{t \mid \forall p \in t. t|_p \in \mathcal{L}\} \; .$$

The next lemma states that the set of languages of trivial automata is closed under the modal operator "globally". On the one hand, this is an interesting closure property, which shows that at least safety properties can be expressed by trivial automata. On the other hand, it is worth looking at the proof of this lemma, as it shows, in a simple setting, all the central ideas that will be used to construct our finitary proof calculus and show its soundness and completeness. The states of the automaton $\mathfrak{A}_{\mathsf{AG}}$ constructed in the proof of Lemma 2.20 should be thought of as annotations proving that $\mathfrak{A}$ has a run starting in various states.

Lemma 2.20. *If $\mathfrak{A}$ is a trivial automaton, then there exist trivial automata $\mathfrak{A}_{\mathsf{EG}}$ and $\mathfrak{A}_{\mathsf{AG}}$ such that $\mathcal{L}(\mathfrak{A}_{\mathsf{EG}}) = \mathsf{EG}\mathcal{L}(\mathfrak{A})$ and $\mathcal{L}(\mathfrak{A}_{\mathsf{AG}}) = \mathsf{AG}\mathcal{L}(\mathfrak{A})$.*

Proof. Roughly speaking, the idea is to construct an alternating automaton that follows one path (for EG) or spawns through all nodes (for AG) and in each step spawns a new automaton that verifies that $\mathfrak{A}$ was a run on the subtree starting at the current node. This alternation can be removed by a simple powerset construction.

Formally, let $\mathfrak{A}$ be given by the state set Q, the initial states I and the transition function δ. Define $Q_{\mathsf{AG}} = \mathfrak{P}(Q)$, $I_{\mathsf{AG}} = \{M \in \mathfrak{P}(Q) \mid M \cap I \neq \emptyset\}$, and

$$\begin{aligned}\delta_{\mathsf{AG}}(M,\mathfrak{f}) = \{(M_1,\ldots,M_{\sharp(\mathfrak{f})},*,\ldots,*) \mid \; & [\, \forall q \in M \exists (q_1,\ldots,q_{\sharp(\mathfrak{f})},*,\ldots,*) \in \delta(\mathfrak{f},q)\\ & \quad q_1 \in M_1 \wedge \ldots \wedge q_{\sharp(\mathfrak{f})} \in M_{\sharp(\mathfrak{f})} \,]\\ & \wedge \forall i (M_i \cap I \neq \emptyset) \; \} \; .\end{aligned}$$

Let $\mathfrak{A}_{\mathsf{AG}}$ be the automaton given by this data. Intuitively, the first condition in the transition function ensures that every state in M can be continued to a run of $\mathfrak{A}$, whereas the second condition ensures that a new run of $\mathfrak{A}$ can be started at every node.

To verify these properties first assume that $t \in \mathsf{AG}\mathcal{L}(\mathfrak{A})$. For every node $p \in t$ set $M_p = \{q \in Q \mid \mathfrak{A}, q \models^\infty t|_p\}$. Then the mapping $p \mapsto M_p$ is a run of $\mathfrak{A}_{\mathsf{AG}}$ on t. The first condition in the transition relation is fulfilled since every state that has a infinite run must be able to make a transition to new states that have an infinite run on the corresponding subtrees. The second condition is satisfied since $t \in \mathsf{AG}\mathcal{L}(\mathfrak{A})$ guarantees that $\mathfrak{A}$ has a run for every subtree; so at every subtree, some initial state has to have a run.

Now assume $t_0 \in \mathcal{L}(\mathfrak{A}_{\mathsf{AG}})$. So there is a run r of $\mathfrak{A}_{\mathsf{AG}}$ on t_0. We have to show that $t_0 \in \mathsf{AG}\mathcal{L}(\mathfrak{A})$. To do so, we show that for *all* trees t, *all* $M \in Q_{\mathsf{AG}}$, if there is *any* run of $\mathfrak{A}_{\mathsf{AG}}$ on t starting in M then for *all* $n \in \mathbb{N}$, it holds that $\forall q \in M. \mathfrak{A}, q \models^n t|_p$.

This indeed shows $t \in \mathsf{AG}\mathcal{L}(\mathfrak{A})$. By the properties of I_{AG} and δ_{AG} we immediately get that for all $p \in t_0$ the set $r(p)$ contains an element $q_p \in I$. Applying the claim to $t_0|_p$ we obtain that $\mathfrak{A}$ has a run, starting in q_p on $t|_p$.

So let us show the claim. We argue by induction on n. For $n = 0$ there's nothing to show. So let $n \geq 1$ and $q \in M$. Assume that t is of the form $t = \mathfrak{f}t_1 \dots t_{\sharp(\mathfrak{f})}$ and let $M_1, \dots, M_{\sharp(\mathfrak{f})}$ the states of the run of $\mathfrak{A}_{\mathsf{AG}}$ at the children the root. Since $(M_1, \dots, M_{\sharp(\mathfrak{f})}, \dots) \in \delta_{\mathsf{AG}}(M, \mathfrak{f})$ there exist $q_1, \dots, q_{\sharp(\mathfrak{f})}$ such that $(q_1, \dots, q_{\sharp(\mathfrak{f})}, \dots) \in \delta(q, \mathfrak{f})$ and $q_i \in M_i$. Applying the induction hypothesis to M_i and t_i we get $\mathfrak{A}, q_i \models^{n-1} t_i$. Together with the transition $q \mapsto (q_1, \dots, q_{\mathfrak{f}})$ we get $\mathfrak{A}, q \models^n t$.

The construction for $\mathfrak{A}_{\mathsf{EG}}$ is similar. □

Taking stock, we see that quite a few safety properties can be expressed by trivial automata. Proposition 2.9 and Lemmata 2.10, 2.11, 2.16, and 2.20 show that the fragment of CTL given by the following grammar can be expressed by trivial automata.

$$\varphi, \psi ::= \mathfrak{f} \mid \varphi \vee \psi \mid \varphi \wedge \psi \mid \mathsf{EX}\varphi \mid \mathsf{AX}\varphi \mid \mathsf{EG}\varphi \mid \mathsf{AG}\varphi$$

Of course $\neg\mathfrak{f}$ can be expressed by an appropriate disjunction over all the other letters of the alphabet.

Even though this grammar probably does not exhaust all the properties expressible by trivial automata, it gives the right flair of the properties being safety properties. We will now show that the simplest liveness property, that is the "eventually" modality, cannot be expressed, not even for word languages.

Definition 2.21 (Word Alphabet)**.** An alphabet Σ is called a *word alphabet*, if all its letters $\mathfrak{f} \in \Sigma$ have arity $\sharp(\mathfrak{f}) = 1$.

Remark 2.22. If Σ is a word alphabet, then the only Σ-terms are ω-words.

Lemma 2.23 (Pumping Lemma for Trivial Automata over Words)**.** *Let $\mathfrak{A}$ be a trivial automaton over a word alphabet Σ. Then there is a natural number n such that for every word w such that $\mathfrak{A} \models^n w$ there is a prefix of w of the form uv with $|uv| \leq n$ and $|v| \geq 1$ such that $uv^\omega \in \mathcal{L}(\mathfrak{A})$.*

Proof. Set $n = |Q| + 1$ where Q is the set of states of $\mathfrak{A}$. Let $w = \mathfrak{f}_0\mathfrak{f}_1\mathfrak{f}_2\dots$ and assume $\mathfrak{A} \models^n w$. Let the states $q_0q_1 \dots q_{n-1}$ constitute such a run up to level n on w. Since $|Q| = n - 1$ there must be $0 \leq i < j < n$ such that $q_i = q_j$. Set $u = \mathfrak{f}_0 \dots \mathfrak{f}_{i-1}$ and $v = \mathfrak{f}_i \dots \mathfrak{f}_{j-1}$. Then $q_0 \dots q_{i-1}(q_i \dots q_{j-1})^\omega$ constitutes a run on uv^ω and u, v are as desired. □

An immediate consequence is, that trivial automata cannot express the property "eventually b", as the following corollary shows.

Corollary 2.24. *The language $\mathcal{L} = a^*b(a+b)^\omega$ is not the language of any trivial automaton.*

Proof. Suppose, for sake of contradiction, that $\mathcal{L} = \mathcal{L}(\mathfrak{A})$ for some trivial automaton $\mathfrak{A}$ and let n be as asserted by Lemma 2.23. Consider $a^nba^\omega \in \mathcal{L} = \mathcal{L}(\mathfrak{A})$ and let u, v be as asserted by the lemma. Since uv is a prefix of a^nba^ω of length at most n, both, u and v must consist of letters a only, and therefore the lemma asserts $a^\omega \in \mathcal{L}(\mathfrak{A}) = \mathcal{L}$ which is not the case. $\square$

3. Infinitary Lambda Trees

Now let Σ' be a fixed set of letters and let $\mathfrak{f}$ from now on only range over elements of Σ'. The choice of the name Σ' will become clear in Definition 5.2, when we have to extend the alphabet in the context of continuous normalisation.

Definition 3.1. The *simple types*, denoted by ρ, σ, τ, are built from the base type ι by arrows $\rho \to \sigma$. The arrow associates to the right. In particular, $\overrightarrow{\rho} \to \iota$ is short for $\rho_1 \to (\rho_2 \to (\ldots(\rho_n \to \iota)\ldots))$.

In the lambda calculus the most common way to from terms is via application. In lambda-trees application is represented by a binary @-node. In linear notation, we omit the "@" and write a tree consisting of an @-node at the root and subtrees s and t just as juxtaposition st. Application associates to the right, i.e., rst is short for $((rs)t)$.

Definition 3.2. The *infinitary simply-typed lambda-trees* over typed terminals Σ' are coinductively given by the grammar

$$r, s ::= x^\rho \mid (\lambda x^\rho t^\sigma)^{\rho\to\sigma} \mid (t^{\rho\to\sigma} s^\rho)^\sigma \mid \mathfrak{f}^{\iota\to\ldots\to\iota\to\iota}\,.$$

In other words, they are, not-necessarily well founded, trees built, in a locally type respecting way, from unary λx^ρ-nodes, binary @-nodes representing application, and leaf nodes consisting of typed variables x^ρ of type ρ and typed constants $\mathfrak{f} \in \Sigma'$ of type $\underbrace{\iota \to \ldots \to \iota}_{\sharp(\mathfrak{f})} \to \iota$.

Here λx^ρ binds free occurrences of the variable x^ρ in its body. Trees with all variables bound are called *closed*.

A lambda-tree with only finitely many non-isomorphic subtrees is called *regular*.

We omit type superscripts if they are clear from the context, or irrelevant.

We usually leave out the words "simply typed", tacitly assuming all our lambda-trees to be simply typed and to use terminals from Σ' only. Figure 2 shows two regular lambda-trees. Arrows are used to show where the pattern repeats, or to draw isomorphic subtrees only once. Note that they denote terms (shown in Figure 1) that are not regular. Here, by "denote" we mean the *term reading* of the normal form.

Remark 3.3. It should be noted that in lambda-trees, as opposed to Σ'-terms, all constants and variables, no matter what their type is, occur at leaf positions.

The reason is, that in a lambda-calculus setting the main concept is that of an application. This is different from first order terms, where the constructors are the main concept. Note that we use lambda-trees to denote Σ'-terms. As these are different concepts, even

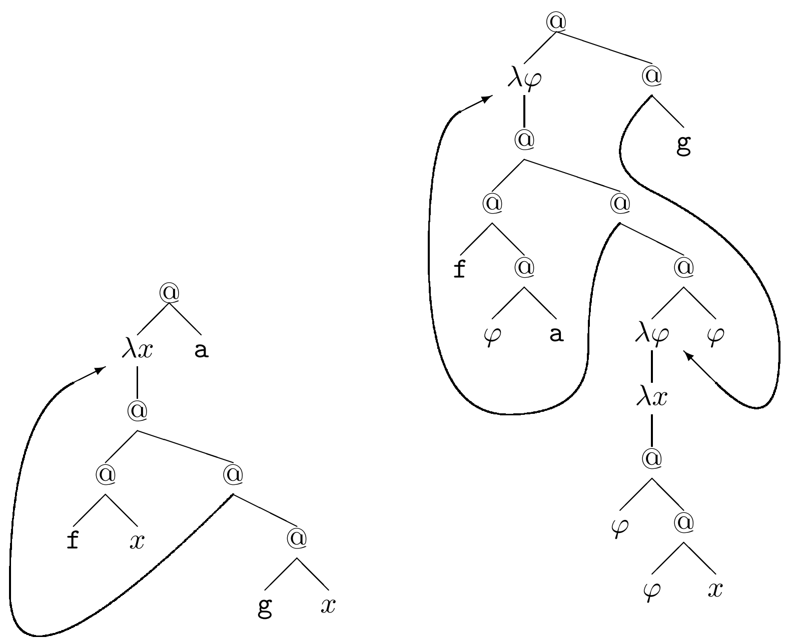
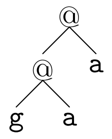
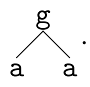


Figure 2: Two regular lambda-trees with denotation being the {f, g, a}-terms in Figure 1.

normal lambda-trees differ from their denotation. For example the lambda-tree 

denotes the Σ'-term 

.

4. Recursion Schemes as Means to Define Regular Lambda Trees

The interest in infinitary lambda-trees in the verification community recently arose by the study of recursion schemes. It could be shown [10, 11] that under a certain "safety" condition the (infinite) terms generated by recursion schemes have decidable monadic second order theory. For our purpose it is enough to consider recursion schemes as a convenient means to define regular lambda-trees.

Definition 4.1. *Recursion schemes* are given by a set of first-order terminal symbols, simply-typed non-terminal symbols and for every non-terminal F an equation

$$F\overrightarrow{x} = e$$

where e is an expression of ground type built up from terminals, non-terminals and the variables $\overrightarrow{x}$ by type-respecting application. There is a distinguished non-terminal symbol S of ground type, called the *start symbol.*

Definition 4.2. Each recursion scheme *denotes*, in the obvious way, a partial, in general infinite, term built from the terminals. Starting from the start symbol, recursively replace the outer-most non-terminals by their definitions with the arguments substituted in appropriately.

$$\begin{array}{lcl} S & = & F\mathtt{a} \\ Fx & = & \mathtt{f}x(F(\mathtt{g}x)) \end{array} \qquad\qquad \begin{array}{lcl} S' & = & F'(W\mathtt{g}) \\ F'\varphi & = & \mathtt{f}(\varphi a)(F'(W\varphi)) \\ W\varphi x & = & \varphi(\varphi x) \end{array}$$

Figure 3: Two recursion schemes.

Definition 4.3. To every recursion scheme is *associated* a regular lambda-tree in the following way. First replace all equations $F\overrightarrow{x} = e$ by

$$F = \lambda\overrightarrow{x}.e$$

where the right hand side is read as a lambda term.

Then, starting from the start symbol, recursively replace all non-terminals by their definition *without performing any computations.*

Remark 4.4. Immediately from the definition we note that the β-normal form of the lambda-tree associated with a recursion scheme, when read a term, *is* the term denoted by that recursion scheme.

Example 4.5. Figure 3 shows two recursion schemes with non-terminals $F\colon \iota \to \iota$, $F'\colon (\iota \to \iota) \to \iota$, $W\colon (\iota \to \iota) \to \iota \to \iota$, and $S, S'\colon \iota$. Their corresponding lambda-trees are the ones shown in Figure 2. The sharing of an isomorphic sub-tree arises as both are translations of the same non-terminal W. As already observed, these recursion schemes denote the terms shown in Figure 1.

Remark 4.6. The notion of a recursion scheme wouldn't change if we allowed λ-abstractions on the right hand side of the equations; we can always build the closure and "factor it out" as a new non-terminal. For example, the $W\varphi$ in the definition of F' in Figure 3 should be thought of as the factored-out closure $(\lambda x.\varphi(\varphi x))$ which is part of a line that originally looked

$$F'\varphi = \mathtt{f}(\varphi a)(F'(\lambda x.\varphi(\varphi x))) \ .$$

5. Continuous Normalisation for the Lambda Calculus

As mentioned in the introduction, we are interested in the question, whether an automaton $\mathfrak{A}$ has a run on the normal form of some lambda-tree t. Our plan to investigate this question is by analysing the term t.

However, there is no bound on the number of nodes of t that have to be inspected, and no bound on the number of beta-reductions to be carried out, before the first symbol of the normal form is determined — if it ever will be. In fact, it may well be that an infinite simply-typed lambda-tree leaves the normal form undefined at some point.

Example 5.1. It should be noted that the typing discipline does not prevent the problem of undefinedness. This is due to inherently infinitary nature of recursion schemes. Let $Y\colon (\iota \to \iota) \to \iota$, $I\colon \iota \to \iota$, and $S\colon \iota$ be non-terminal symbols and consider the recursion scheme

$$\begin{array}{lcl} Y\varphi & = & \varphi(Y\varphi) \\ Ix & = & x \\ S & = & YI \end{array}$$

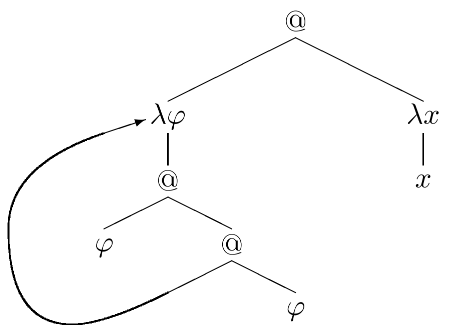


Figure 4: The lambda-tree associated to the recursion scheme in Example 5.1.

with start symbol S.

Computing the normal form of the associated lambda-tree gives the following infinite reduction sequence $S = YI \to_\beta I(YI) \to_\beta YI \to_\beta \ldots$. Of course, the fact that the computation will never produce a terminal symbol can, in this example, also be trivially seen from the fact that the whole recursion scheme does not contain any terminal symbol.

Whereas the unboundedness of the number of symbols to be inspected is merely a huge inconvenience, the possibility of undefinedness makes it unclear what it even is supposed to mean that "$\mathfrak{A}$ has a run on the normal form of t" — if there is no such normal form.

This problem of possible undefinedness of the normal form is similar to a situation in proof theory, where only strong principles guarantee the termination of the cut-elimination procedure, whereas the operation itself can be defined in primitive recursive arithmetic. Continuous Normalisation was introduced by Mints [12, 14] in order to separate cut-elimination for semiformal systems from their ordinal analysis. The operational aspects of normalisation, i.e., the manipulations on infinitary derivations, are isolated and described independently of the system's proof theoretic complexity, but at the expense of introducing the void logical rule

$$(\mathcal{R})\,\frac{\Gamma}{\Gamma}$$

of repetition. Note that this rule is both, logically valid and has the subformula property.

Using the repetition rule, the cut-elimination operator becomes primitive recursive and can be studied in its own right. As Mints observed, this cut-elimination operator can also be applied to non-wellfounded derivations, resulting in a continuous function on derivation trees (a concise exposition can be found in an article [6] by Buchholz).

The possibility to handle infinite computations is particularly natural in the realm of the lambda calculus, where non-termination actually does happen. Let us explain the idea of continuous normalisation for the lambda-calculus [2, 3] by considering the recursion scheme in Example 5.1. The associated lambda tree is shown in Figure 4.

We look at the outer-most constructor of the term and see an application. Just from this knowledge we cannot deduce any constructor of the normal form. The normal form *read as a lambda-tree* could be an application as well, e.g., if the left term is a terminal; since we're trying to compute the normal form as a Σ'-tree, even in this case we would have to inspect the term further to find out which terminal it is, the term starts with. But, more importantly, it could also be that the left term is a λ-abstraction, in which a beta-reduction has to be carried out and the normal form could look almost arbitrary. So we don't know any constructor of the normal form yet. On the other hand, we want to be uniformly continuous with identity as modulus of continuity; in other words, we want to ensure that the output of all nodes of level k only depend on the input of level k. We solve this problem by outputting $\mathcal{R}$, signalling that we have to read more input to decide what the normal form will look like.

Having output $\mathcal{R}$ we now may look at the next level of the term. Seeing the $\lambda\varphi$ we still don't any constructor of the normal form, but at least we know that we have to wait for a different reason — we have to carry out some computation. Therefore we output a β constructor, signalling that the delay in the output is due to a beta-reduction being carried out. Note that in a certain sense (made precise in Lemma 5.4) this β "justifies" the first $\mathcal{R}$-constructor. The application we have seen in the first step has disappeared due to the beta-reduction being carried out. A different form of justification would be outputting a Σ'-term, where the lambda-tree reading contains an application. For example the term $\mathtt{f}\mathtt{a}$ with $\mathtt{f}$ and $\mathtt{a}$ both terminals would have continuous normal form $\mathcal{R}(\mathtt{f}(\mathtt{a}))$, with the $\mathcal{R}$ justified by the fact that $\mathtt{f}$ is applied to one argument $\mathtt{a}$.

After this beta-reduction the term $I(YI)$ is remaining, so we're looking at an application again, and, as before, wait by saying $\mathcal{R}$. Again, there is a lambda abstraction to the left of the application, so we say β and carry out the reduction due to the λx, leaving us with YI, which happens to be the term we started with. Of course, we don't know this yet, as the only thing we see so far is the outermost @. But the fact that we arrived at YI again ensures that the pattern $\mathcal{R}\beta\mathcal{R}\beta\dots$ of the normal form will repeat.

Let us now formally introduce continuous normalisation. As mentioned, we extend the language by two new terminals. The $\mathcal{R}$-constructor for a delay due to inspection of an application and the β-constructor for a delay due to a beta-reduction.

Definition 5.2. Define $\Sigma = \Sigma' \cup \{\mathcal{R}, \beta\}$ with $\mathcal{R}, \beta$ two new terminals of arity one.

The continuous normalisation procedure, which will compute the continuous normal form, follows the informal description above. In other words, if we see an application we output $\mathcal{R}$ and carry on by reading more input. If we see a lambda-abstraction our typing restrictions force that we have to have collected some arguments before, so that a beta-reduction has to be carried out, accompanied by a β constructor; in the more general case [2] of the untyped lambda calculus [5] we would have to do a case distinction on whether we have at least one argument collected or not. In the latter case the normal form would start with a λ. Finally, if we find a terminal symbol we construct a term, which is the terminal symbol applied to the continuous normal forms of the arguments collected so far.

In our official Definition 5.3 of the continuous normal form, the expression $t\underline{@}\overrightarrow{t}$ should be read as "the continuous normal form of t, with arguments $t_1, \dots, t_n$ collected already". Correspondingly the continuous normal form of t is $t\underline{@}()$ which we also abbreviate by t^{β}.

Definition 5.3. For t, $\overrightarrow{t}$ closed infinitary simply-typed lambda-trees such that $t\overrightarrow{t}$ is of ground type we define a Σ-term $t\underline{@}\overrightarrow{t}$ coinductively as follows.

$$\begin{array}{ll}
(rs)\underline{@}\overrightarrow{t} & = \mathcal{R}(r\underline{@}(s, \overrightarrow{t})) \\
(\lambda x.r)\underline{@}(s, \overrightarrow{t}) & = \beta(r[s/x]\underline{@}\overrightarrow{t}) \\
\mathfrak{f}\underline{@}\overrightarrow{t} & = \mathfrak{f}(t_1^{\beta}, \dots, t_n^{\beta})
\end{array}$$

Here we used $r[s/x]$ to denote the substitution of s for x in r. This substitution is necessarily capture free as s is closed. By $\mathfrak{f}(T_1, \dots, T_n)$ we denote the term with label $\mathfrak{f}$ at the root and $T_1, \dots, T_n$ as its n children; this includes the case $n = 0$, where $\mathfrak{f}()$ denotes the term consisting of a single node $\mathfrak{f}$. Similar notation is used for $\mathcal{R}(T)$ and $\beta(T)$. Moreover we used r^β as a shorthand for $r\underline{@}()$.

The term t^β is also called the *continuous normal form* of t.

A first observation is that the definition obeys the informal idea of "justifying" the delay constructors. We note that, whenever the number of collected arguments increases we output a $\mathcal{R}$, and whenever the number of arguments decreases (due to an argument being consumed by a beta-reduction) we output a β. This bookkeeping of the number of collected arguments is made precise in the next lemma.

Lemma 5.4. *If* $t\underline{@}(t_1, \dots, t_k) = \mathcal{W}_1(\mathcal{W}_2(\dots(\mathcal{W}_\ell.\mathfrak{f}(\overrightarrow{s}))))$ *with* $\mathcal{W}_1, \dots, \mathcal{W}_\ell \in \{\mathcal{R}, \beta\}$ *then the equation* $k + |\{i \mid \mathcal{W}_i = \mathcal{R}\}| = |\{i \mid \mathcal{W}_i = \beta\}| + \sharp(\mathfrak{f})$ *holds.*

Proof. A simple induction on ℓ. If $\ell = 0$, the claim $k = \sharp(\mathfrak{f})$ follows from the typing requirements. Note that we allowed the expression $t\underline{@}\overrightarrow{t}$ only of $t\overrightarrow{t}$ is well typed of ground type. If $\ell > 0$ we distinguish whether t is an application or a lambda-abstraction. In either case we unfold the definition of $t\underline{@}\overrightarrow{t}$ once and can apply the induction hypothesis. □

Next we will study the relation between lambda terms, their continuous normal forms, and their normal forms in the usual sense, in case the latter exists. This, on the one hand, will give a clearer picture on what the continuous normal form of a lambda term is. On the other hand, it will also justify the claim, that is not only technically more convenient for the development in the rest of this article to use continuous normalisation, but that it is also more informative.

As an immediate observation, the reader might note that any property expressible by some automaton $\mathfrak{A}$ working on Σ'-trees can be lifted to a property on Σ-trees by "ignoring the additional $\mathcal{R}$ and β constructors". The lifted property can also be expressed by an automaton. We just have to extend the transition function δ by setting $\delta(q, \mathcal{R}) = \delta(q, \beta) = \{(q, *, \dots, *)\}$. In particular, using continuous normalisation does not cause any disadvantages for the decision problem we are interested in.

We already mentioned that output up to depth h only depends on the input up to depth h. To make this idea precise, we first define a notion of similarity for lambda-tree or Σ-terms. The relation $r \approx_k s$ holds, if r and s coincide up to level k. This is made precise in the following definition.

Definition 5.5. For Σ-terms r, s we define, by induction on k, the relation $r \approx_k s$ by the following rules.

$$\frac{}{r \approx_0 s} \qquad \frac{r_1 \approx_k s_1, \ \dots, \ r_\ell \approx_k s_\ell}{\mathfrak{f}(r_1, \dots, r_k) \approx_{k+1} \mathfrak{f}(s_1, \dots, s_\ell)}$$

For lambda-trees r, s we define, by induction on k, the relation $r \approx_k s$ by the following rules.

$$\frac{}{r \approx_0 s} \qquad \frac{r \approx_k s}{\lambda x.r \approx_{k+1} \lambda x.s} \qquad \frac{r \approx_k r' \qquad s \approx_k s'}{rs \approx_{k+1} r's'}$$

$$\frac{}{x \approx_k x} \qquad \frac{}{\mathfrak{f} \approx_k \mathfrak{f}}$$

Proposition 5.6. *If* r *and* s *are both* Σ*-terms or both lambda-trees and* $\ell, k \in \mathbb{N}$*, then* $r \approx_\ell s$ *and* $\ell \geq k$ *imply* $r \approx_k s$*.*

Proof. Induction on k. □

Remark 5.7. Obviously, $s = t$ holds if and only if $\forall k.s \approx_k t$. Moreover, each of the relations $\approx_k$ is an equivalence relation.

Proposition 5.6 and Remark 5.7 together show, that we obtain a metric d if we set $d(s,t)$ to be 0, if $s = t$ and otherwise set $d(s,t) = \frac{1}{k+1}$ where k is maximal such that $s \approx_k t$. We will now show that continuous normalisation is continuous with respect to this topology. In fact, we even show a stronger statement of uniform continuity.

Proposition 5.8. *If $s \approx_k s'$ and $t_1 \approx_k t'_1$, ..., $t_n \approx_k t'_n$ then $s \underline{@} \overrightarrow{t} \approx_k s' \underline{@} \overrightarrow{t}'$.*

Proof. Induction on k. If $k = 0$, there is nothing to show. If $k > 0$, then the outermost constructors of s and s' have to coincide. We unfold the definitions of $s \underline{@} \overrightarrow{t}$ and $s' \underline{@} \overrightarrow{t}'$ once and apply the induction hypothesis. □

Now that we know (by Proposition 5.8) that continuous normalisation does not consume too much input in order to produce the output, we aim at showing that the output is actually useful and not just a pointless collection of delay constructors. We have already seen (in Lemma 5.4) that the $\mathcal{R}$ constructors are justified by either β constructors or the arity of the terminals in the output produced. So what remains to show is, that the β constructors are not arbitrary, but in a reasonable sense related to the underlying computation. In fact, it will turn out, that every β constructor corresponds to a beta reduction in the head normalisation strategy; compare Lemmata 5.9 and 5.10. It is well known that this reduction strategy finds a normal form, if there is one.

Lemma 5.9. *If $t \underline{@} \overrightarrow{t} = \mathcal{W}_1(\ldots(\mathcal{W}_k(\mathfrak{f}(s_1,\ldots,s_{\sharp(\mathfrak{f})}))))$ with $\mathcal{W}_i \in \{\mathcal{R},\beta\}$ then there are lambda-trees $r_1,\ldots,r_{\sharp(\mathfrak{f})}$ such that*

- *$t\overrightarrow{t}$ reduces in n head-reduction steps to $\mathfrak{f}\overrightarrow{r}$ where n is the number of β constructors, i.e., $n = |\{i \mid \mathcal{W}_i = \beta\}|$, and*
- *for each i it holds that $r_i^\beta = s_i$.*

Proof. Induction on k. If $k = 0$, inspection of Definition 5.3 of $t \underline{@} \overrightarrow{t}$ shows that it must be the case that $t = \mathfrak{f}$. So, in this case $\mathfrak{f} \underline{@} \overrightarrow{t} = \mathfrak{f}(t_1^\beta,\ldots,t_{\sharp(\mathfrak{f})}^\beta)$ and we can take $\overrightarrow{r}$ to be $\overrightarrow{t}$.

If $k > 0$ and $\mathcal{W}_1 = \beta$ it must be the case that $t = \lambda x.t'$. Then $(\lambda x.t') \underline{@} (t_1,t_2,\ldots,t_\ell) = \beta((t'[t_1/x]) \underline{@} (t_2,\ldots,t_\ell))$. So $(t'[t_1/x]) \underline{@} (t_2,\ldots,t_\ell) = \mathcal{W}_2(\ldots(\mathcal{W}_k(\mathfrak{f}(s_1,\ldots,s_{\sharp(\mathfrak{f})}))))$ and the induction hypothesis gives us $\overrightarrow{r}$ with $r_i^\beta = s_i$ such that $t\overrightarrow{t}$ reduces in $n-1$ steps to $\mathfrak{f}\overrightarrow{r}$. Since, moreover, in one head reduction step, $t\overrightarrow{t} = (\lambda x.t')t_1t_2\ldots t_\ell$ reduces to $t'[t_1/x]t_2\ldots t_\ell$, this yields the claim. If $k > 0$ and $\mathcal{W}_1 = \mathcal{R}$ the claim is immediate from the induction hypothesis. □

Lemma 5.10. *If $t\overrightarrow{t}$ reduces by n head reduction steps to $\mathfrak{f}r_1\ldots r_{\sharp(\mathfrak{f})}$ then for some $\mathcal{W}_1,\ldots,\mathcal{W}_k \in \{\mathcal{R},\beta\}$ with $|\{i \mid \mathcal{W}_i = \beta\}| = n$ we have $t \underline{@} \overrightarrow{t} = \mathcal{W}_1(\ldots(\mathcal{W}_k(\mathfrak{f}(r_1^\beta,\ldots,r_{\sharp(\mathfrak{f})}^\beta))))$.*

Proof. Induction on n. If $n = 0$ then $t\overrightarrow{t}$ must be of the form $\mathfrak{f}\overrightarrow{r}$ and, indeed, $t \underline{@} \overrightarrow{t} = \mathcal{R}(\ldots(\mathcal{R}(\mathfrak{f} \underline{@} \overrightarrow{r}))) = \mathcal{R}(\ldots(\mathcal{R}(\mathfrak{f}(r_1^\beta,\ldots,r_{\sharp(\mathfrak{f})}^\beta))))$.

If $n > 0$ then t is of the form $(\lambda xs)\overrightarrow{s}$. Writing $\overrightarrow{t}'$ for $\overrightarrow{s}\overrightarrow{t}$ we note that $t \underline{@} \overrightarrow{t} = \mathcal{R}(\ldots(\mathcal{R}((\lambda x.s) \underline{@} \overrightarrow{t}'))) = \mathcal{R}(\ldots(\mathcal{R}(\beta(s[t'_1/x] \underline{@} (t'_2,\ldots,t'_\ell)))))$. Since the head reduct of $t\overrightarrow{t}$ is $s[t'_1/x]t'_2\ldots t'_\ell$, the induction hypothesis yields the claim. □

It should be noted that in the special case of $\overrightarrow{t}$ being the empty list, Lemmata 5.9 and 5.10 talk about the continuous normal form of t.

6. Finitary Semantics and Proof System

Let $\mathfrak{A}$ be a fixed nondeterministic tree automaton with state set Q and transition function $\delta\colon Q \times \Sigma \to \mathfrak{P}((Q \cup \{*\})^N)$. The main technical idea of this article is to use a finite semantics for the simple types, describing how $\mathfrak{A}$ "sees" an object of that type.

Definition 6.1. For τ a simple type we define $[\![\tau]\!]$ inductively as follows.

$$\begin{array}{ll} [\![\iota]\!] & = \mathfrak{P}(Q) \\ [\![\rho \to \sigma]\!] & = {}^{[\![\rho]\!]}[\![\sigma]\!] \end{array}$$

In other words, we start with the power set of the state set of $\mathfrak{A}$ in the base case, and use the full set theoretic function space for arrow-types.

Remark 6.2. Obviously all the $[\![\tau]\!]$ are finite sets.

Example 6.3. Taking $\mathfrak{A}$ to be the automaton of Example 2.7, we have $[\![\iota]\!] = \{\emptyset, \{q_2\}, \{q_1\}, Q\}$ and examples of elements of $[\![\iota \to \iota]\!]$ include the identity function id, as well as the "swap function" swap defined by $\mathrm{swap}(\emptyset) = \emptyset$, $\mathrm{swap}(Q) = Q$, $\mathrm{swap}(\{q_2\}) = \{q_1\}$, and $\mathrm{swap}(\{q_1\}) = \{q_2\}$.

Definition 6.4. $[\![\tau]\!]$ is partially ordered as follows.
- For $R, S \in [\![\iota]\!]$ we set $R \sqsubseteq S$ iff $R \subseteq S$.
- For $f, g \in [\![\rho \to \sigma]\!]$ we set $f \sqsubseteq g$ iff $\forall a \in [\![\rho]\!]. fa \sqsubseteq ga$.

Remark 6.5. Obviously suprema and infima with respect to $\sqsubseteq$ exist.

We often need the concept "continue with f after reading one $\mathcal{R}$ symbol". We call this $\mathcal{R}$-lifting. Similar for β.

Definition 6.6. For $f \in [\![\overrightarrow{\rho} \to \iota]\!]$ we define the liftings $\mathcal{R}(f), \beta(f) \in [\![\overrightarrow{\rho} \to \iota]\!]$ as follows.

$$\begin{array}{rcl} \mathcal{R}(f)(\overrightarrow{a}) & = & \{q \mid \delta(q, \mathcal{R}) \cap f\overrightarrow{a} \times \{*\} \times \ldots \times \{*\} \neq \emptyset\} \\ \beta(f)(\overrightarrow{a}) & = & \{q \mid \delta(q, \beta) \cap f\overrightarrow{a} \times \{*\} \times \ldots \times \{*\} \neq \emptyset\} \end{array}$$

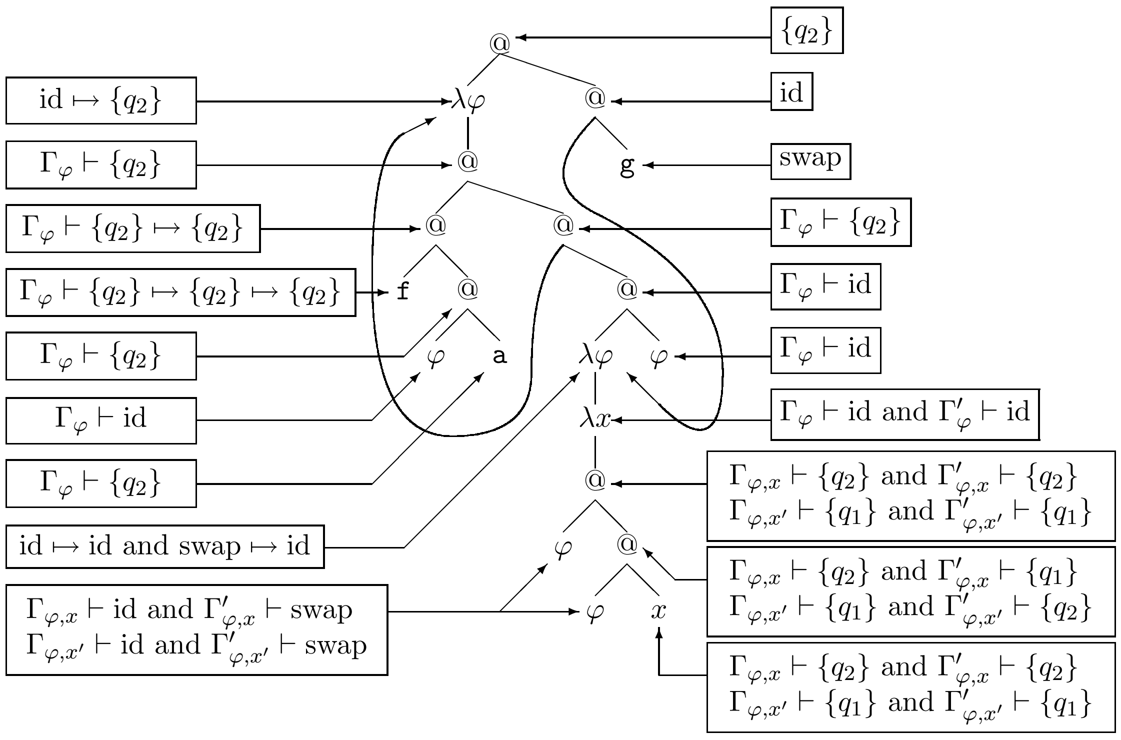
Remark 6.7. If $\mathfrak{A}$ is obtained from an automaton working on Σ'-terms by setting $\delta(q, \mathcal{R}) = \delta(q, \beta) = \{(q, *, \ldots, *)\}$ then $\mathcal{R}(f) = \beta(f) = f$ for all f.

Using this finite semantics we can use it to annotate a lambda-tree by semantical values for its subtrees to show that the denoted term has good properties with respect to $\mathfrak{A}$. We start by an example.

Example 6.8. The second recursion scheme in Figure 3 denotes a term where the "side branches" contain $2, 4, 8, \ldots, 2^n, \ldots$ times the letter g. As these are all even numbers, the automaton $\mathfrak{A}$ of Example 2.7 should have a run starting in q_2.

We now informally argue how a formal "proof" of this fact can be obtained by assigning semantical values to the nodes of the corresponding lambda-tree, which is the right tree in Figure 2. The notion of "proof" will be made formal in Definition 6.10.

So we start by assigning the root $\{q_2\} \in [\![\iota]\!]$. Since the term is an application, we have to guess the semantics of the argument (of type $\iota \to \iota$). Our (correct) guess is, that it keeps the parity of gs unchanged, hence our guess is id; the function side then must be



$\{q_2\}$
id
swap
$\mathrm{id} \mapsto \{q_2\}$
$\Gamma_\varphi \vdash \{q_2\}$
$\Gamma_\varphi \vdash \{q_2\} \mapsto \{q_2\}$
$\Gamma_\varphi \vdash \{q_2\} \mapsto \{q_2\} \mapsto \{q_2\}$
$\Gamma_\varphi \vdash \{q_2\}$
$\Gamma_\varphi \vdash \mathrm{id}$
$\Gamma_\varphi \vdash \{q_2\}$
$\mathrm{id} \mapsto \mathrm{id}$ and $\mathrm{swap} \mapsto \mathrm{id}$
$\Gamma_{\varphi,x} \vdash \mathrm{id}$ and $\Gamma'_{\varphi,x} \vdash \mathrm{swap}$
$\Gamma_{\varphi,x'} \vdash \mathrm{id}$ and $\Gamma'_{\varphi,x'} \vdash \mathrm{swap}$
$\Gamma_\varphi \vdash \{q_2\}$
$\Gamma_\varphi \vdash \mathrm{id}$
$\Gamma_\varphi \vdash \mathrm{id}$
$\Gamma_\varphi \vdash \mathrm{id}$ and $\Gamma'_\varphi \vdash \mathrm{id}$
$\Gamma_{\varphi,x} \vdash \{q_2\}$ and $\Gamma'_{\varphi,x} \vdash \{q_2\}$
$\Gamma_{\varphi,x'} \vdash \{q_1\}$ and $\Gamma'_{\varphi,x'} \vdash \{q_1\}$
$\Gamma_{\varphi,x} \vdash \{q_2\}$ and $\Gamma'_{\varphi,x} \vdash \{q_1\}$
$\Gamma_{\varphi,x'} \vdash \{q_1\}$ and $\Gamma'_{\varphi,x'} \vdash \{q_2\}$
$\Gamma_{\varphi,x} \vdash \{q_2\}$ and $\Gamma'_{\varphi,x} \vdash \{q_2\}$
$\Gamma_{\varphi,x'} \vdash \{q_1\}$ and $\Gamma'_{\varphi,x'} \vdash \{q_1\}$
@
$\lambda\varphi$
@
@
g
@
@
f
@
@
φ
a
$\lambda\varphi$
φ
λx
@
φ
@
φ
x

Figure 5: A proof that $\mathfrak{A}$ has an infinite run starting in q_2 on the denoted term.

something that maps id to $\{q_2\}$. Let us denote by $\mathrm{id} \mapsto \{q_2\}$ the function in ${}^{[\iota \to \iota]}[\iota]$ defined by $(\mathrm{id} \mapsto \{q_2\})(\mathrm{id}) = \{q_2\}$ and $(\mathrm{id} \mapsto \{q_2\})(f) = \emptyset$ if $f \neq \mathrm{id}$.

The next node to the left is an abstraction. So we have to assign the body the value $\{q_2\}$ in a context where φ is mapped to id. Let us denote this context by Γ_φ.

In a similar way we fill out the remaining annotations. Figure 5 shows the whole proof. Here Γ'_φ is the context that maps φ to swap; moreover $\Gamma_{\varphi,x}$, $\Gamma'_{\varphi,x}$, $\Gamma_{\varphi,x'}$, and $\Gamma'_{\varphi,x'}$ are the same as Γ_φ and Γ'_φ but with x mapped to $\{q_2\}$ and $\{q_1\}$, respectively.

It should be noted that a similar attempt to assign semantical values to the other lambda-tree in Figure 2 fails at the down-most x where in the context Γ with $\Gamma(x) = \{q_2\}$ we cannot assign x the value $\{q_1\}$.

To make the intuition of the example precise, we formally define a "proof system" of possible annotations (Γ, a) for a (sub)tree. Since the $[\![\tau]\!]$ are all finite sets, there are only finitely many possible annotations.

To simplify the later argument of our proof, which otherwise would be coinductive, we add a level n to our notion of proof. This level should be interpreted as "for up to n steps we can pretend to have a proof". This reflects the fact that coinduction is nothing but induction on observations.

Definition 6.9. A *context* is a finite mapping from variables x^σ to their corresponding semantics $[\![\sigma]\!]$. We use Γ to range over contexts.

If Γ is a context, x a variable of type σ and $a \in [\![\sigma]\!]$ we denote by Γ^a_x the context Γ modified in that x is mapped to a, regardless of whether x was or was not in the domain of Γ.

Definition 6.10. For Γ a context, $a \in [\![\rho]\!]$ a value, and t an infinitary, maybe open, lambda-tree of type ρ, with free variables among $\mathrm{dom}(\Gamma)$, we define

$$\Gamma \vdash^n_{\mathfrak{A}} a \sqsubseteq t : \rho$$

by induction on the natural number n as follows.

- $\Gamma \vdash^0_{\mathfrak{A}} a \sqsubseteq t : \rho$ always holds.
- $\Gamma \vdash^n_{\mathfrak{A}} a \sqsubseteq x_i : \rho$ holds, provided $a \sqsubseteq \Gamma(x_i)$.
- $\Gamma \vdash^{n+1}_{\mathfrak{A}} a \sqsubseteq st : \sigma$ holds, provided there exists $f \in [\![\rho \to \sigma]\!]$, $u \in [\![\rho]\!]$ such that $a \sqsubseteq \mathcal{R}(fu)$, $\Gamma \vdash^n_{\mathfrak{A}} f \sqsubseteq s : \rho \to \sigma$, and $\Gamma \vdash^n_{\mathfrak{A}} u \sqsubseteq t : \rho$.
- $\Gamma \vdash^{n+1}_{\mathfrak{A}} f \sqsubseteq \lambda x^\rho.s : \rho \to \sigma$ holds, provided for all $a \in [\![\rho]\!]$ there is a $b_a \in [\![\sigma]\!]$ such that $fa \sqsubseteq \beta(b_a)$ and $\Gamma^a_x \vdash^n_{\mathfrak{A}} b_a \sqsubseteq s : \sigma$.
- $\Gamma \vdash^n_{\mathfrak{A}} f \sqsubseteq \mathfrak{f} : \iota \to \ldots \to \iota \to \iota$ holds, provided for all $\overrightarrow{a} \in [\![\overrightarrow{\iota}]\!]$ we have $f\overrightarrow{a} \subset \{q \mid \delta(q, \mathfrak{f}) \cap a_1 \times \ldots \times a_{\sharp(\mathfrak{f})} \times \{*\} \times \ldots \times \{*\} \neq \emptyset\}$.

It should be noted that all the quantifiers in the rules range over finite sets. Hence the correctness of a rule application can be checked effectively (and even by a finite automaton).

We write $\Gamma \vdash^\infty_{\mathfrak{A}} a \sqsubseteq t : \rho$ to denote $\forall n. \Gamma \vdash^n_{\mathfrak{A}} a \sqsubseteq t : \rho$.

Remark 6.11. Obviously $\Gamma \vdash^{n+1}_{\mathfrak{A}} a \sqsubseteq t : \rho$ implies $\Gamma \vdash^n_{\mathfrak{A}} a \sqsubseteq t : \rho$. Moreover, $a' \sqsubseteq a$ and $\Gamma \vdash^n_{\mathfrak{A}} a \sqsubseteq t : \rho$ imply $\Gamma \vdash^n_{\mathfrak{A}} a' \sqsubseteq t : \rho$. Finally, $\Gamma \vdash^n_{\mathfrak{A}} a \sqsubseteq t : \rho$, if $\Gamma' \vdash^n_{\mathfrak{A}} a \sqsubseteq t : \rho$ for some Γ' which agrees with Γ on the free variables of t.

Also, in the second an in the last clause we may assume without loss of generality, that $n > 0$. However, this assumption is not necessary, and it is even technically more convenient not to do so.

Remark 6.12. We notice that the proof informally given in Example 6.8 and shown in Figure 5 complies with the formal Definition 6.10. Indeed, the annotations shown in the figure are valid for any n.

As already mentioned, for t a term with finitely many free variables, the annotations (Γ, a) come from a fixed finite set, since we can restrict Γ to the set of free variables of t. If, moreover, t has only finitely many different sub-trees, that is to say, if t is regular, then only finitely many terms t have to be considered. So we obtain

Proposition 6.13. *For t regular, it is decidable whether* $\Gamma \vdash^\infty_{\mathfrak{A}} a \sqsubseteq t : \rho$.

Before we continue and show our calculus to be sound (Section 7) and complete (Section 8) let us step back and see what we will then have achieved, once our calculus is proven sound and complete.

Proposition 6.13 gives us decidability for terms denoted by regular lambda-trees, and hence in particular for trees obtained by recursion schemes. Moreover, since the annotations only have to fit locally, individual subtrees of the lambda-tree can be verified separately. This is of interest, as for each non-terminal a separate subtree is generated. In other words, this approach allows for modular verification; think of the different non-terminals as different subroutines. As the semantics is the set-theoretic one, the annotations are clear enough to be meaningful, if we have chosen our automaton in such a way that the individual states can be interpreted extensionally, for example as "even" versus "odd" number of gs.

It should also be noted, that the number of possible annotations only depends on the type of the subtree, and on $\mathfrak{A}$, that is, the property to be investigated. Fixing $\mathfrak{A}$ and the allowed types (which both usually tend to be quite small), the amount of work to be carried out grows only linearly with the representation of t as a regular lambda-tree. For every node we have to make a guess and we have to check whether this guess is consistent with the guesses for the (at most two) child nodes. Given that the number of nodes of the representation of t grows linearly with the size of the recursion scheme, the problem is in fixed-parameter-$\mathcal{NP}$, which doesn't seem too bad for practical applications.

7. Truth Relation and Proof of Soundness

The soundness of a calculus is usually shown by using a logical relation, that is, a relation indexed by a type that interprets the type arrow "$\rightarrow$" as logical arrow "$\Rightarrow$"; in other words, we define partial truth predicates for the individual types [17].

Since we want to do induction on the "observation depth" n of our proof $\cdot \vdash^n_{\mathfrak{A}} \cdot \sqsubseteq \cdot : \tau$ we have to include that depth in the definition of our truth predicates $\cdot \prec\!\!\!\prec^n_{\mathfrak{A}} \cdot : \tau$. For technical reasons we have to build in weakening on this depth as well.

Definition 7.1. For $f \in [\![\overrightarrow{\rho} \to \iota]\!]$, $n \in \mathbb{N}$, t a closed infinitary lambda tree of type $\overrightarrow{\rho} \to \iota$, the relation $f \prec\!\!\!\prec^n_{\mathfrak{A}} t : \overrightarrow{\rho} \to \iota$ is defined by induction on the type as follows.

$$\begin{array}{l} f \prec\!\!\!\prec^n_{\mathfrak{A}} t : \overrightarrow{\rho} \to \iota \qquad \text{iff} \\ \forall \ell \leq n \forall \overrightarrow{a} \in [\![\overrightarrow{\rho}]\!] \forall \overrightarrow{r} : \overrightarrow{\rho} \\ \qquad (\forall i.\ a_i \prec\!\!\!\prec^\ell_{\mathfrak{A}} r_i : \rho_i) \Rightarrow \forall q \in f\overrightarrow{a}.\ \mathfrak{A}, q \models^\ell t@\overrightarrow{r} \end{array}$$

Remark 7.2. Immediately from the definition we get the following monotonicity property. If $f \sqsubseteq f'$ and $f' \prec\!\!\!\prec^n_{\mathfrak{A}} t : \rho$ then $f \prec\!\!\!\prec^n_{\mathfrak{A}} t : \rho$.

Remark 7.3. In the special case $\overrightarrow{\rho} = \varepsilon$ we get

$$S \prec\!\!\!\prec^n_{\mathfrak{A}} t : \iota \qquad \text{iff} \qquad \forall q \in S. \mathfrak{A}, q \models^n t^\beta$$

Here we used that $\forall \ell \leq n. \mathfrak{A}, q \models^\ell s$ iff $\mathfrak{A}, q \models^n s$.

Immediately from the definition we obtain weakening in the level.

Proposition 7.4. *If $f \prec\!\!\!\prec^n_{\mathfrak{A}} t : \rho$ then $f \prec\!\!\!\prec^{n-1}_{\mathfrak{A}} t : \rho$.*

Theorem 7.5. *Assume $\Gamma \vdash^n_{\mathfrak{A}} a \sqsubseteq t : \rho$ for some Γ with domain $\{x_1, \ldots, x_2\}$. For all $\ell \leq n$ and all closed terms $\overrightarrow{t} : \overrightarrow{\rho}$, if $\forall i.\ \Gamma(x_i) \prec\!\!\!\prec^\ell_{\mathfrak{A}} t_i : \rho_i$ then $a \prec\!\!\!\prec^\ell_{\mathfrak{A}} t[\overrightarrow{t}/\overrightarrow{x}] : \rho$.*

Proof. Induction on n, cases according to $\Gamma \vdash^n_{\mathfrak{A}} a \sqsubseteq t : \rho$.

- Case $\Gamma \vdash^0_{\mathfrak{A}} a \sqsubseteq t : \rho$ always. Use that $a \prec\!\!\!\prec^0_{\mathfrak{A}} \ldots : \rho$ holds always.
- Case $\Gamma \vdash^n_{\mathfrak{A}} a \sqsubseteq x_i : \rho$ because of $a \sqsubseteq \Gamma(x_i)$.

  Assume $\forall i. \Gamma(x_i) \prec\!\!\!\prec^\ell_{\mathfrak{A}} t_i : \rho_i$. We have to show $a \prec\!\!\!\prec^\ell_{\mathfrak{A}} \underbrace{x_i[\overrightarrow{t}/\overrightarrow{x}]}_{t_i} : \rho$, which follows from one of our assumptions by Remark 7.2.
- Case $\Gamma \vdash^{n+1}_{\mathfrak{A}} a \sqsubseteq st : \sigma$ thanks to $f \in [\![\rho \to \sigma]\!]$, $u \in [\![\rho]\!]$ such that $a \sqsubseteq \mathcal{R}(fu)$, $\Gamma \vdash^n_{\mathfrak{A}} f \sqsubseteq s : \rho \to \sigma$, and $\Gamma \vdash^n_{\mathfrak{A}} u \sqsubseteq t : \rho$.

  Let $\ell \leq n+1$ be given, and $\overrightarrow{t} : \overrightarrow{\rho}$ such that $\forall i.\ \Gamma(x_i) \prec\!\!\!\prec^\ell_{\mathfrak{A}} t_i : \rho_i$. We have to show $a \prec\!\!\!\prec^\ell_{\mathfrak{A}} (st) \underbrace{[\overrightarrow{t}/\overrightarrow{x}]}_{\eta} : \sigma$.

  Let σ have the form $\sigma = \overrightarrow{\sigma} \to \iota$. Let $k \leq \ell$ be given and $\overrightarrow{s} : \overrightarrow{\sigma}$, $c_i \in [\![\sigma_i]\!]$ such that $c_i \prec\!\!\!\prec^k_{\mathfrak{A}} s_i : \sigma_i$. We have to show for all $q \in a\overrightarrow{c}$ that $\mathfrak{A}, q \models^k \underbrace{(s\eta t\eta)@\overrightarrow{r}}_{\mathcal{R}.(s\eta @(t\eta, \overrightarrow{r}))}$.

Hence it suffices to show that there is a $\tilde{q} \in \delta(q, \mathcal{R})$ such that $\mathfrak{A}, \tilde{q} \models^{k-1} s\eta\underline{@}(t\eta, \overrightarrow{r})$.

Since $k \leq \ell \leq n+1$, we have $k-1 \leq n$. Using Proposition 7.4 various times we obtain $\forall i.\ \Gamma(x_i) \prec\!\!\prec_{\mathfrak{A}}^{k-1} t_i : \rho_i$. Hence we may use the induction hypotheses to $\Gamma \vdash_{\mathfrak{A}}^{n} f \sqsubseteq s : \rho \to \sigma$ and obtain $f \prec\!\!\prec_{\mathfrak{A}}^{k-1} s\eta : \rho \to \sigma$. Applying the induction to $\Gamma \vdash_{\mathfrak{A}}^{n} u \sqsubseteq t : \rho$ yields $u \prec\!\!\prec_{\mathfrak{A}}^{k-1} t\eta : \rho$.

Applying Proposition 7.4 to $c_i \prec\!\!\prec_{\mathfrak{A}}^{k} s_i : \sigma_i$ yields $c_i \prec\!\!\prec_{\mathfrak{A}}^{k-1} s_i : \sigma_i$. Therefore $\forall \hat{q} \in fu\overrightarrow{c}.\ \mathfrak{A}, \hat{q} \models^{k-1} s\eta\underline{@}(t\eta, \overrightarrow{r})$.

Since $a \sqsubseteq \mathcal{R}(fu)$ we get $\forall q \in a\overrightarrow{c}\,\exists \tilde{q} \in \delta(q, \mathcal{R}).\ \tilde{q} \in fu\overrightarrow{c}$. This together with the last statement yields the claim.

- *Case* $\Gamma \vdash_{\mathfrak{A}}^{n+1} f \sqsubseteq \lambda x^{\rho}.s : \rho \to \sigma$ thanks to $\forall a \in [\![\rho]\!]\ \exists b_a \in [\![\sigma]\!]$ such that $fa \sqsubseteq \beta(b_a)$ and $\Gamma^{a}_{x} \vdash_{\mathfrak{A}}^{n} b_a \sqsubseteq s : \sigma$.

  Let $\ell \leq n+1$ be given and $\overrightarrow{t} : \overrightarrow{\rho}$ with $\Gamma(x_i) \prec\!\!\prec_{\mathfrak{A}}^{\ell} t_i : \rho_i$.

  We have to show $f \prec\!\!\prec_{\mathfrak{A}}^{\ell} (\lambda x^{\rho} s^{\sigma})\eta : \rho \to \sigma$ where η is short for $[\overrightarrow{t}/\overrightarrow{x}]$.

  Let σ have the form $\sigma = \overrightarrow{\sigma} \to \iota$. Let $k \leq \ell$ be given and $r : \rho$, $\overrightarrow{s} : \overrightarrow{\sigma}$, $c \in [\![\rho]\!]$, $c_i \in [\![\sigma_i]\!]$ such that $c \prec\!\!\prec_{\mathfrak{A}}^{k} r : \rho$, $c_i \prec\!\!\prec_{\mathfrak{A}}^{k} s_i : \sigma_i$. We have to show for all $q \in fc\overrightarrow{c}$ that $\mathfrak{A}, q \models^{k} \underbrace{(\lambda x s)\eta\underline{@}(r, \overrightarrow{s})}_{\beta.s\eta^{r}_{x}\underline{@}\overrightarrow{s}}$.

  Hence it suffices to show that there is a $\tilde{q} \in \delta(q, \beta)$ such that $\mathfrak{A}, \tilde{q} \models^{k-1} s\eta^{r}_{x}\underline{@}\overrightarrow{s}$.

  We know $c \prec\!\!\prec_{\mathfrak{A}}^{k} r : \rho$; using Proposition 7.4 we get $c \prec\!\!\prec_{\mathfrak{A}}^{k-1} r : \rho$ and $\forall i.\ \Gamma(x_i) \prec\!\!\prec_{\mathfrak{A}}^{k-1} t_i : \rho_i$. Since $k \leq \ell \leq n+1$ we get $k-1 \leq n$, hence we may apply the induction hypothesis to $\Gamma^{a}_{x} \vdash_{\mathfrak{A}}^{n} b_a \sqsubseteq s : \sigma$ and obtain $b_a \prec\!\!\prec_{\mathfrak{A}}^{k-1} s\eta^{r}_{x} : \sigma$.

  Since again by Proposition 7.4 we also know $c_i \prec\!\!\prec_{\mathfrak{A}}^{k-1} s_i : \sigma_i$, we obtain for all $\hat{q} \in b_a\overrightarrow{c}$ that $\mathfrak{A}, \hat{q} \models^{k-1} s\eta^{r}_{x}\underline{@}\overrightarrow{s}$.

  Since $fc \sqsubseteq \beta(b_c)$ we get that $\forall q \in fc\overrightarrow{c}\,\exists \tilde{q} \in \delta(q, \beta).\ \tilde{q} \in b_c\overrightarrow{c}$. This, together with the last statement yields the claim.

- Case $\Gamma \vdash_{\mathfrak{A}}^{n} f \sqsubseteq \mathfrak{f} : \iota \to \iota$ thanks to $\forall \overrightarrow{a} \in [\![\iota]\!].\ f\overrightarrow{a} \subset \{q \mid \delta(q, \mathfrak{f}) \cap \overrightarrow{a} \neq \emptyset\}$.

  Let $\ell \leq n$ be given and $\overrightarrow{t} : \overrightarrow{\rho}$ such that $\forall i.\ \Gamma(x_i) \prec\!\!\prec_{\mathfrak{A}}^{\ell} t_i : \rho_i$. We have to show $f \prec\!\!\prec_{\mathfrak{A}}^{\ell} \underbrace{\mathfrak{f}[\overrightarrow{t}/\overrightarrow{x}]}_{\mathfrak{f}} : \iota \to \iota$.

  Let $k \leq \ell$ be given and $\overrightarrow{r} : \overrightarrow{\iota}$, $\overrightarrow{S} \in [\![\iota]\!]$ such that $S_i \prec\!\!\prec_{\mathfrak{A}}^{\ell} r_i : \iota$. We have to show for all $q \in f\overrightarrow{S}$ that $\mathfrak{A}, q \models^{\ell} \underbrace{\mathfrak{f}\underline{@}\overrightarrow{r}}_{\mathfrak{f}\overrightarrow{r}^{\beta}}$.

  From $S_i \prec\!\!\prec_{\mathfrak{A}}^{\ell} r_i : \iota$ we get $\forall \tilde{q}_i \in S_i.\ \mathfrak{A}, \tilde{q}_i \models^{\ell} r_i^{\beta}$. Hence the claim follows since $\forall q \in f\overrightarrow{S}\ \exists \overrightarrow{\tilde{q}} \in \delta(a, \mathfrak{f}).\ \overrightarrow{\tilde{q}} \in \overrightarrow{S}$. □

It should be noted that in the proof of Theorem 7.5 in the cases of the λ-rule and the application-rule it was possible to use the induction hypothesis due to the fact that we used *continuous* normalisation, as opposed to standard normalisation.

Corollary 7.6. *For t a closed infinitary lambda term we get immediately from Theorem 7.5*

$$\emptyset \vdash_{\mathfrak{A}}^{n} S \sqsubseteq t : \iota \quad \Longrightarrow \quad \forall q \in S.\ \mathfrak{A}, q \models^{n} t^{\beta}$$

In particular, if $\emptyset \vdash_{\mathfrak{A}}^{\infty} S \sqsubseteq t : \iota$ then $\forall q \in S.\ \mathfrak{A}, q \models^{\infty} t^{\beta}$.

8. The Canonical Semantics and the Proof of Completeness

If we want to prove that there is an infinite run, then, in the case of an application st, we have to guess a value for the term t "cut out".

We could assume an actual run be given and analyse the "communication", in the sense of game semantics [9], between the function s and its argument t. However, it is simpler to assign each term a "canonical semantics" $\langle\!\langle t\rangle\!\rangle_{\mathfrak{A}\infty}$, roughly the supremum of all values we have canonical proofs for.

The subscript ∞ signifies that we only consider infinite runs. The reason is that the level n in our proofs $\Gamma \vdash^n_{\mathfrak{A}} a \sqsubseteq t : \rho$ is not a tight bound; whenever we have a proofs of level n, then there are runs for at least n steps, but on the other hand, runs might be longer than the maximal level of a proof. This is due to the fact that β-reduction moves subterms "downwards", that is, further away from the root, and in that way may construct longer runs. The estimates in our proof calculus, however, have to consider (in order to be sound) the worst case, that is, that an argument is used immediately.

Since, in general, the term t may also have free variables, we have to consider a canonical semantics $\langle\!\langle t\rangle\!\rangle^\Gamma_{\mathfrak{A}\infty}$ with respect to an environment Γ.

Definition 8.1. By induction on the type we define for t a closed infinite lambda-tree of type $\rho = \overrightarrow{\rho} \to \iota$ its canonical semantics $\langle\!\langle t\rangle\!\rangle_{\mathfrak{A}\infty} \in [\![\rho]\!]$ as follows.

$$\langle\!\langle t\rangle\!\rangle_{\mathfrak{A}\infty}(\overrightarrow{a}) = \{q \mid \exists \overrightarrow{s} : \overrightarrow{\rho}\ .\ \langle\!\langle \overrightarrow{s}\rangle\!\rangle_{\mathfrak{A}\infty} \sqsubseteq \overrightarrow{a}\ \wedge \mathfrak{A}, q \models^\infty t\underline{@}\overrightarrow{s}\}$$

Remark 8.2. For t a closed term of base type we have $\langle\!\langle t\rangle\!\rangle_{\mathfrak{A}\infty} = \{q \mid \mathfrak{A}, q \models^\infty t^\beta\}$.

Definition 8.3. For Γ a context, $t\colon \rho$ typed in context Γ of type $\rho = \overrightarrow{\rho} \to \iota$ we define $\langle\!\langle t\rangle\!\rangle^\Gamma_{\mathfrak{A}\infty} \in [\![\rho]\!]$ by the following explicit definition.

$$\begin{aligned}\langle\!\langle t\rangle\!\rangle^\Gamma_{\mathfrak{A}\infty}(\overrightarrow{a}) = \{q \mid\ & \exists \eta.\ \mathrm{dom}(\eta) = \mathrm{dom}(\Gamma) \wedge \\ & (\forall x \in \mathrm{dom}(\Gamma).\eta(x) \text{ closed} \wedge \langle\!\langle \eta(x)\rangle\!\rangle_{\mathfrak{A}\infty} \sqsubseteq \Gamma(x)) \wedge \\ & \exists \overrightarrow{s} : \overrightarrow{\rho}.\langle\!\langle \overrightarrow{s}\rangle\!\rangle_{\mathfrak{A}\infty} \sqsubseteq \overrightarrow{a}\ \wedge\ \mathfrak{A}, q \models^\infty t\eta\underline{@}\overrightarrow{s}\}\end{aligned}$$

Remark 8.4. For t a closed term and $\Gamma = \emptyset$ we have $\langle\!\langle t\rangle\!\rangle^\Gamma_{\mathfrak{A}\infty} = \langle\!\langle t\rangle\!\rangle_{\mathfrak{A}\infty}$.

Proposition 8.5. *If s has type $\overrightarrow{\sigma} \to \iota$ in some context compatible with Γ, and η is some substitution with $\mathrm{dom}(\eta) = \mathrm{dom}(\Gamma)$ such that for all $x \in \mathrm{dom}(\Gamma)$ we have $\eta(x)$ closed and $\langle\!\langle \eta(x)\rangle\!\rangle_{\mathfrak{A}\infty} \sqsubseteq \Gamma(x)$, then*

$$\langle\!\langle s\eta\rangle\!\rangle_{\mathfrak{A}\infty} \sqsubseteq \langle\!\langle s\rangle\!\rangle^\Gamma_{\mathfrak{A}\infty}$$

Proof. Let $\overrightarrow{a} \in [\![\overrightarrow{\sigma}]\!]$ and $q \in \langle\!\langle s\eta\rangle\!\rangle_{\mathfrak{A}\infty}(\overrightarrow{a})$ be given. Then there are $\overrightarrow{s} : \overrightarrow{\sigma}$ with $\langle\!\langle \overrightarrow{s}\rangle\!\rangle_{\mathfrak{A}\infty} \sqsubseteq \overrightarrow{a}$ such that $\mathfrak{A}, q \models^\infty s\eta\underline{@}\overrightarrow{s}$. Together with the assumed properties of η this witnesses $q \in \langle\!\langle s\rangle\!\rangle^\Gamma_{\mathfrak{A}\infty}(\overrightarrow{a})$. ☐

Lemma 8.6. *If r and s are terms of type $\sigma \to \overrightarrow{\rho} \to \iota$ and σ, respectively, in some context compatible with Γ, then we have*

$$\langle\!\langle rs\rangle\!\rangle^\Gamma_{\mathfrak{A}\infty} \sqsubseteq \mathcal{R}(\langle\!\langle r\rangle\!\rangle^\Gamma_{\mathfrak{A}\infty}\langle\!\langle s\rangle\!\rangle^\Gamma_{\mathfrak{A}\infty})$$

Proof. Let $\overrightarrow{a} \in [\![\overrightarrow{\rho}]\!]$ and $q \in \langle\!\langle rs\rangle\!\rangle^\Gamma_{\mathfrak{A}\infty}(\overrightarrow{a})$ be given. Then there is η with $\forall x \in \mathrm{dom}(\Gamma).\ \langle\!\langle \eta(x)\rangle\!\rangle_{\mathfrak{A}\infty} \sqsubseteq \Gamma(x)$ and there are $\overrightarrow{s} : \overrightarrow{\rho}$ with $\langle\!\langle \overrightarrow{s}\rangle\!\rangle_{\mathfrak{A}\infty} \sqsubseteq \overrightarrow{a}$ and

$$\mathfrak{A}, q \models^\infty \underbrace{(rs)\eta\underline{@}\overrightarrow{s}}_{\mathcal{R}.r\eta\underline{@}(s\eta,\overrightarrow{s})}$$

Hence there is a $q' \in \delta(q, \mathcal{R})$ with $\mathfrak{A}, q' \models^\infty r\eta \underline{@}(s\eta, \overrightarrow{s})$. It suffices to show that for this q' we have $q' \in \langle\!\langle r\rangle\!\rangle^\Gamma_{\mathfrak{A}\infty}\langle\!\langle s\rangle\!\rangle^\Gamma_{\mathfrak{A}\infty}\overrightarrow{a}$.

By Proposition 8.5 we have $\langle\!\langle s\eta\rangle\!\rangle_{\mathfrak{A}\infty} \sqsubseteq \langle\!\langle s\rangle\!\rangle^\Gamma_{\mathfrak{A}\infty}$ and we already have $\langle\!\langle \overrightarrow{s}\rangle\!\rangle_{\mathfrak{A}\infty} \sqsubseteq \overrightarrow{a}$. So the given η together with $s\eta$ and $\overrightarrow{s}$ witnesses $q' \in \langle\!\langle r\rangle\!\rangle^\Gamma_{\mathfrak{A}\infty}\langle\!\langle s\rangle\!\rangle^\Gamma_{\mathfrak{A}\infty}\overrightarrow{a}$. ☐

Lemma 8.7. *Assume that $\lambda x.r$ has type $\sigma \to \overrightarrow{\rho} \to \iota$ in some context compatible with Γ. Then*

$$\langle\!\langle \lambda xr\rangle\!\rangle^\Gamma_{\mathfrak{A}\infty}(a) \sqsubseteq \beta(\langle\!\langle r\rangle\!\rangle^{\Gamma^a_x}_{\mathfrak{A}\infty})$$

Proof. Let $\overrightarrow{a} \in [\![\overrightarrow{\rho}]\!]$ and $q \in \langle\!\langle \lambda xr\rangle\!\rangle^\Gamma_{\mathfrak{A}\infty}(a, \overrightarrow{a})$ be given. Then there is an η with $\forall x \in \mathrm{dom}(\Gamma)$ we have $\eta(x)$ closed and $\langle\!\langle \eta(x)\rangle\!\rangle_{\mathfrak{A}\infty} \sqsubseteq \Gamma(x)$ and there are $s, \overrightarrow{s}$ with $\langle\!\langle s\rangle\!\rangle_{\mathfrak{A}\infty} \sqsubseteq a$ and $\langle\!\langle \overrightarrow{s}\rangle\!\rangle_{\mathfrak{A}\infty} \sqsubseteq \overrightarrow{a}$ such that

$$\mathfrak{A}, q \models^\infty \underbrace{(\lambda xr)\eta\underline{@}(s, \overrightarrow{s})}_{\beta.r_x[s]\eta\underline{@}\overrightarrow{s}}$$

So there is a $\tilde{q} \in \delta(q, \beta)$ with $\mathfrak{A}, \tilde{q} \models^\infty r_x[s]\eta\underline{@}\overrightarrow{s}$. It suffices to show that $\tilde{q} \in \langle\!\langle r\rangle\!\rangle^{\Gamma^a_x}_{\mathfrak{A}\infty}(\overrightarrow{a})$.

By the properties of η and since $\langle\!\langle s\rangle\!\rangle_{\mathfrak{A}\infty} \sqsubseteq a$ we know that for all $y \in \mathrm{dom}(\Gamma^a_x)$ we have $\langle\!\langle \eta(y)\rangle\!\rangle_{\mathfrak{A}\infty} \sqsubseteq \Gamma^a_x(y)$. This witnesses $\tilde{q} \in \langle\!\langle r\rangle\!\rangle^{\Gamma^a_x}_{\mathfrak{A}\infty}(\overrightarrow{a})$. ☐

Lemma 8.8. $\langle\!\langle x\rangle\!\rangle^\Gamma_{\mathfrak{A}\infty} \sqsubseteq \Gamma(x)$

Proof. Assume x of type $\overrightarrow{\rho} \to \iota$, let $\overrightarrow{a} \in [\![\overrightarrow{\rho}]\!]$ and $q \in \langle\!\langle x\rangle\!\rangle^\Gamma_{\mathfrak{A}\infty}(\overrightarrow{a})$ be given. We have to show $\Gamma(x)(\overrightarrow{a})$.

Since $q \in \langle\!\langle x\rangle\!\rangle^\Gamma_{\mathfrak{A}\infty}(\overrightarrow{a})$, there is η with $\eta(x) \sqsubseteq a$ and $\overrightarrow{s}\colon \overrightarrow{\rho}$ with $\langle\!\langle \overrightarrow{s}\rangle\!\rangle^\Gamma_{\mathfrak{A}\infty} \sqsubseteq \overrightarrow{a}$ and $\mathfrak{A}, q \models^\infty \underbrace{x\eta}_{\eta(x)}\underline{@}\overrightarrow{s}$.

But then $\overrightarrow{s}$ witness that $q \in \langle\!\langle \eta(x)\rangle\!\rangle_{\mathfrak{A}\infty}(\overrightarrow{a}) \subset \Gamma(x)(\overrightarrow{a})$ where the last subset relation holds since $\langle\!\langle \eta(x)\rangle\!\rangle_{\mathfrak{A}\infty} \sqsubseteq \Gamma(x)$. ☐

Theorem 8.9. $\Gamma \vdash^n_{\mathfrak{A}} \langle\!\langle t\rangle\!\rangle^\Gamma_{\mathfrak{A}\infty} \sqsubseteq t : \rho$

Proof. Induction on n, cases on t. Trivial for $n = 0$. So let $n > 0$. We distinguish cases according to t

- Case rs^σ. By induction hypothesis $\Gamma \vdash^{n-1}_{\mathfrak{A}} \langle\!\langle r\rangle\!\rangle^\Gamma_{\mathfrak{A}\infty} \sqsubseteq r : \sigma \to \rho$ and $\Gamma \vdash^{n-1}_{\mathfrak{A}} \langle\!\langle s\rangle\!\rangle^\Gamma_{\mathfrak{A}\infty} \sqsubseteq s : \sigma$. Moreover, by Lemma 8.6 $\langle\!\langle rs\rangle\!\rangle^\Gamma_{\mathfrak{A}\infty} \sqsubseteq \mathcal{R}(\langle\!\langle r\rangle\!\rangle^\Gamma_{\mathfrak{A}\infty}\langle\!\langle s\rangle\!\rangle^\Gamma_{\mathfrak{A}\infty})$. Hence $\Gamma \vdash^n_{\mathfrak{A}} \langle\!\langle rs\rangle\!\rangle^\Gamma_{\mathfrak{A}\infty} \sqsubseteq rs : \rho$.
- Case $\lambda x^\sigma r$. By induction hypothesis we have for all $a \in [\![\sigma]\!]$ that $\Gamma^a_x \vdash^{n-1}_{\mathfrak{A}} \langle\!\langle r\rangle\!\rangle^{\Gamma^a_x}_{\mathfrak{A}\infty} \sqsubseteq r : \rho$. By Lemma 8.7 we have $\langle\!\langle \lambda xr\rangle\!\rangle^\Gamma_{\mathfrak{A}\infty}(a) \sqsubseteq \beta(\langle\!\langle r\rangle\!\rangle^{\Gamma^a_x}_{\mathfrak{A}\infty})$.
  Hence $\Gamma \vdash^n_{\mathfrak{A}} \langle\!\langle \lambda xr\rangle\!\rangle^\Gamma_{\mathfrak{A}\infty} \sqsubseteq \lambda xr : \sigma \to \rho$.
- Case x. By Lemma 8.8 we have $\langle\!\langle x\rangle\!\rangle^\Gamma_{\mathfrak{A}\infty} \sqsubseteq \Gamma(x)$ and hence $\Gamma \vdash^n_{\mathfrak{A}} \langle\!\langle x\rangle\!\rangle^\Gamma_{\mathfrak{A}\infty} \sqsubseteq x : \rho$.
- Case $t = \mathfrak{f}$ a terminal symbol. We have to show $\Gamma \vdash^n_{\mathfrak{A}} \langle\!\langle \mathfrak{f}\rangle\!\rangle^\Gamma_{\mathfrak{A}\infty} \sqsubseteq \mathfrak{f} : \iota \to \iota$.
  So, let $\overrightarrow{S} \in [\![\overrightarrow{\iota}]\!]$ and $q \in \langle\!\langle \mathfrak{f}\rangle\!\rangle^\Gamma_{\mathfrak{A}\infty}(S)$. Hence there are $\overrightarrow{s}$ of type ι with $\langle\!\langle s_i\rangle\!\rangle_{\mathfrak{A}\infty} \sqsubseteq S_i$ and $\mathfrak{A}, q \models^\infty \underbrace{\mathfrak{f}\underline{@}\overrightarrow{s}}_{\mathfrak{f}(\overrightarrow{s^\beta})}$.
  So there is $(\tilde{q}_1, \dots, \tilde{q}_{\sharp(\mathfrak{f})}, *, \dots, *) \in \delta(q, \mathfrak{f})$ with $\mathfrak{A}, \tilde{q}_i \models^\infty s_i^\beta$. But then $\tilde{q}_i \in \langle\!\langle s_i\rangle\!\rangle_{\mathfrak{A}\infty} \subset S_i$. ☐

Corollary 8.10. *If $t\colon \iota$ is closed and of ground type then $\emptyset \vdash^n_{\mathfrak{A}} \{q \mid \mathfrak{A}, q \models^\infty t^\beta\} \sqsubseteq t : \iota$.*

Proof. By Remarks 8.4 and 8.2 we have $\langle\!\langle t\rangle\!\rangle^\emptyset_{\mathfrak{A}\infty} = \langle\!\langle t\rangle\!\rangle_{\mathfrak{A}\infty} = \{q \mid \mathfrak{A}, q \models^\infty t^\beta\}$. So the claim follows from Theorem 8.9. ☐

Finally, let us sum up what we have achieved.

Corollary 8.11. *For t a closed regular lambda term, and $q_0 \in Q$ it is decidable whether $\mathfrak{A}, q_0 \models^\infty t^\beta$.*

Proof. By Proposition 6.13 it suffices to show that $\emptyset \vdash^\infty_{\mathfrak{A}} \{q_0\} \sqsubseteq t : \iota$ holds, if and only if $\mathfrak{A}, q_0 \models^\infty t^\beta$.

The "if"-direction follows from Corollary 8.10 and the weakening provided by Remark 6.11. The "only if"-direction is provided by Corollary 7.6. □

Note that, since there are only finitely many ways to extend a proof of level n to a proof of level $n+1$ and all proofs of level $n+1$ come from a proof of level n the corollary implies, by König's Lemma, that $\mathfrak{A}, q \models^\infty t^\beta$ implies $\emptyset \vdash^\infty_{\mathfrak{A}} \{q\} \sqsubseteq t : \iota$.

9. Model Checking

Theorem 9.1. *Given a tree $\mathcal{T}$ defined by an arbitrary recursion scheme (of arbitrary level) and a property φ expressible by a trivial automaton, it is decidable whether $\mathcal{T} \models \varphi$.*

Proof. Let t be the infinite lambda-tree associated with the recursion scheme. Then t is effectively given as a regular closed lambda term of ground type and $\mathcal{T}$ is the normal form of t.

Let $\mathfrak{A}_\varphi$ be the automaton (with initial state q_0) describing φ. By keeping the state when reading a $\mathcal{R}$ or β it can be effectively extended to an automaton $\mathfrak{A}$ that works on the continuous normal form, rather than on the usual one. So $\mathcal{T} \models \varphi \Leftrightarrow \mathfrak{A}, q_0 \models^\infty t^\beta$. The latter, however, is decidable by Corollary 8.11. □

Remark 9.2. As shown in Section 2, the above theorem is in particular applicable to CTL-properties built from letters, conjunction, disjunction, "next", and "globally".

Remark 9.3. As discussed after Proposition 6.13 the complexity is fixed-parameter nondeterministic linear time in the size of the recursion scheme, if we consider φ and the allowed types as a parameter.

Finally, looking back at the technical development, it is not clear to the author, whether this approach can be extended in a smooth way to work for arbitrary automata, as opposed to only trivial ones. It is tempting to conjecture that appropriate annotations of the proofs with priorities could extend the concept to parity automata (and hence the full of Monadic Second Order). However, all the ways that seemed obvious to the author failed.

One technical problem is that several paths might lead to the same state at the same node, but with different priorities visited so far. A more fundamental problem is the way the runs are constructed in the proofs throughout this article; we're given a run by induction hypothesis and add a move at its *beginning.* As all acceptance conditions ignore finite prefixes, all the promises to visit some state eventually are pushed in the future indefinitely. So, some promise on how long it will take for some promised event to happen seems to be needed in the annotations, at least if we want these global conditions to fit with our local arguments. It is not clear to the author whether and how this can be achieved.

References

[1] K. Aehlig. A finite semantics of simply-typed lambda terms for infinite runs of automata. In Z. Esik, editor, *Procedings of the 20th international Workshop on Computer Science Logic (CSL '06)*, volume 4207 of *Lecture Notes in Computer Science*, pages 104–118. Springer Verlag, Sept. 2006.

[2] K. Aehlig and F. Joachimski. On continuous normalization. In *Proceedings of the Annual Conference of the European Association for Computer Science Logic (CSL '02)*, volume 2471 of *Lecture Notes in Computer Science*, pages 59–73. Springer Verlag, 2002.

[3] K. Aehlig and F. Joachimski. Continuous normalization for the lambda-calculus and Gödel's T. *Annals of Pure and Applied Logic*, 133(1–3):39–71, May 2005.

[4] K. Aehlig, J. G. de Miranda, and C. H. L. Ong. The monadic second order theory of trees given by arbitrary level-two recursion schemes is decidable. In P. Urzyczyn, editor, *Proceedings of the 7th International Conference on Typed Lambda Calculi and Applications (TLCA '05)*, volume 3461 of *Lecture Notes in Computer Science*, pages 39–54. Springer-Verlag, Apr. 2005.

[5] H. Barendregt. The type free lambda calculus. In J. Barwise, editor, *Handbook of Mathematical Logic*, volume 90 of *Studies in Logic and the Foundations of Mathematics*, chapter D.7, pages 1091–1132. North-Holland Publishing Company, 1977.

[6] W. Buchholz. Notation systems for infinitary derivations. *Archive for Mathematical Logic*, 30:277–296, 1991.

[7] D. Caucal. On infinite transition graphs having a decidable monadic theory. In F. Meyer auf der Heide and B. Monien, editors, *Proceedings of the 23th International Colloquium on Automata, Languages and Programming (ICALP '96)*, volume 1099 of *Lecture Notes in Computer Science*, pages 194–205. Springer Verlag, 1996.

[8] B. Courcelle. The monadic second-order logic of graphs IX: Machines and their behaviours. *Theoretical Comput. Sci.*, 151(1):125–162, 1995.

[9] J. M. E. Hyland and C.-H. L. Ong. On full abstraction for PCF. *Information and Computation*, 163(2):285–408, Dec. 2000.

[10] T. Knapik, D. Niwiński, and P. Urzyczyn. Deciding monadic theories of hyperalgebraic trees. In S. Abramsky, editor, *Proceedings of the 5th International Conference on Typed Lambda Caculi and Applications (TLCA '01)*, volume 2044 of *Lecture Notes in Computer Science*, pages 253–267. Springer Verlag, 2001.

[11] T. Knapik, D. Niwiński, and P. Urzyczyn. Higher-order pushdown trees are easy. In M. Nielson, editor, *Proceedings of the 5th International Conference Foundations of Software Science and Computation Structures (FOSSACS '02)*, volume 2303 of *Lecture Notes in Computer Science*, pages 205–222, Apr. 2002.

[12] G. Kreisel, G. E. Mints, and S. G. Simpson. The use of abstract language in elementary metamathematics: Some pedagogic examples. In R. Parikh, editor, *Logic Colloquium*, volume 453 of *Lecture Notes in Mathematics*, pages 38–131. Springer Verlag, 1975.

[13] O. Kupferman and M. Y. Vardi. An automata-theoretic approach to reasoning about infinite-state systems. In E. A. Emerson and A. P. Sistla, editors, *12th International Conference on Computer Aided Verification (CAV '00)*, volume 1855 of *Lecture Notes in Computer Science*, pages 36–52. Springer Verlag, 2000.

[14] G. E. Mints. Finite investigations of transfinite derivations. *Journal of Soviet Mathematics*, 10:548–596, 1978. Translated from: Zap. Nauchn. Semin. LOMI 49 (1975). Cited after Grigori Mints. *Selected papers in Proof Theory*. Studies in Proof Theory. Bibliopolis, 1992.

[15] C.-H. L. Ong. On model-checking trees generated by higher-order recursion schemes. In *Proceedings of the Twenty Frist Annual IEEE Symposium on Logic in Computer Science (LICS '06)*, pages 81–90, 2006.

[16] M. O. Rabin. Decidability of second-order theories and automata on infinite trees. *Transactions of the American Mathematical Society*, 141:1–35, July 1969.

[17] W. W. Tait. Intensional interpretations of functionals of finite type. *The Journal of Symbolic Logic*, 32(2):198–212, 1967.

[18] I. Walukiewicz. Pushdown processes: Games and model-checking. *Information and Computation*, 164(2):234–263, Jan. 2001.