Functional Reachability

C.-H. L. Ong
University of Oxford

N. Tzevelekos
University of Oxford

***Abstract*—What is *reachability* in higher-order functional programs? We formulate reachability as a decision problem in the setting of the prototypical functional language PCF, and show that even in the recursion-free fragment generated from a finite base type, several versions of the reachability problem are undecidable from order 4 onwards, and several other versions are reducible to each other. We characterise a version of the reachability problem in terms of a new class of tree automata introduced by Stirling at FoSSaCS 2009, called *Alternating Dependency Tree Automata* (ADTA). As a corollary, we prove that the ADTA non-emptiness problem is undecidable, thus resolving an open problem raised by Stirling. However, by restricting to contexts constructible from a finite set of variable names, we show that the corresponding solution set of a given instance of the reachability problem is regular. Hence the relativised reachability problem is decidable.**

I. Introduction

In the simplest form, Reachability is the decision problem: Given a state of a state-transition system (e.g. an error state paired with a program point), is it reachable from the start state? Reachability testing has had a major impact in software model checking; it is now a standard approach to checking safety properties in industry. In the past decade, great strides have been made in model-checking reachability of first-order (recursive) procedural programs. Tools such as SLAM [1] and Blast [7] showcase the remarkable achievements of the computer-aided verification community in the engineering of scalable software model checkers. Perhaps surprisingly no reachability checker has yet been developed for *higher-order* programming languages such as ML, Ocaml, Haskell, F#. Indeed, to our knowledge, reachability of higher-order functional computation *per se* does not appear to have been studied in the literature. We initiate just such an investigation here.

Reachability of higher-order functional programs is quite different from that of first-order imperative programs. Functional programs are state-less, and it is unclear what their program points are (because the term being evaluated is being changed by substitution as the computation unfolds). Further, functional reachability is *contextual*: the flow of control within a (higher-order, open) term should be analysed in relation to all its *program contexts*; it is thus much more complex than graph reachability, which is what first-order reachability boils down to.

Consider the following decision problem in the rather purified setting of PCF, generated from a *finite* base type o.

> CONTEXTUAL REACHABILITY: Given a PCF term M of type A and a subterm N^α with occurrence α, is there a program context $C[\text{-}]$ such that $C[M]$ is a program (i.e. closed term of type o) and the evaluation of $C[M]$ causes control to flow to N^α?

Our starting point is the question: Is CONTEXTUAL REACHABILITY decidable? A precise (and equivalent) way to formulate the problem is to replace the subterm N^α in M by a distinguished error constant $\star$ — call the resultant term $M^\star$ — and ask if there is a PCF program-context $C[\text{-}]$ such that $C[M^\star]$ evaluates to $\star$. Here we regard the *principal term* $M^\star$ and the *context* $C[\text{-}]$ as elements of a larger language, PCF*, which is PCF augmented with $\star$, with evaluation rules so extended as to propagate $\star$ to the top.

More generally, consider the following parameterised decision problem, where the (closed) principal term ranges over $\mathcal{L}_1$, the (applicative) context ranges over $\mathcal{L}_2$, both $\mathcal{L}_1$ and $\mathcal{L}_2$ are sublanguages of PCF*, and θ ranges over the base type $o := \{\mathsf{t}, \mathsf{f}, \star\}$.

> θ-REACH $[\mathcal{L}_1, \mathcal{L}_2]$: Given a closed $\mathcal{L}_1$-term $M : A_1 \to \cdots \to A_n \to o$, are there closed $\mathcal{L}_2$-terms $N_1, \cdots, N_n$ such that $M\overline{N}$ evaluates to θ?

We can formulate the preceding reachability problem (equivalently) as $\star$-REACH $[\text{PCF}^\star, \text{PCF}]$.

For a sharper analysis, we consider the *finitary* (i.e. recursion-free) sublanguages, fPCF* and fPCF. Note that "divergence" is definable in PCF (e.g. $\mathbf{Y}_o(\lambda x.x)$) but not in fPCF. Thus we also consider fPCF$_\perp$, fPCF augmented with a divergence constant $\perp$. We obtain two results.

(i) *Undecidability*. By exploiting (the key lemma behind) Loader's proof of the undecidability of PCF observational equivalence [13], we show that CONTEXTUAL REACHABILITY, $\star$-REACH $[\text{PCF}^\star, \text{PCF}]$, t-REACH $[\text{fPCF}_\perp, \text{fPCF}]$ and t-REACH $[\text{fPCF}^\star, \text{fPCF}]$ (and several others) are all undecidable from order 4 onwards.

(ii) *Equivalence*. The problems $\star$-REACH $[\text{fPCF}^\star, \text{fPCF}]$ and $\perp$-REACH $[\text{fPCF}_\perp, \text{fPCF}]$ are polynomially reducible to each other. Whether they are decidable is open.

Motivated by the open problem, we analyse fPCF* computation automata-theoretically. Stirling [22] has recently introduced a new kind of tree automata called *Alternating Dependency Tree Automata* (ADTA) which are an accepting device for trees with a binding relation called Σ-*binding trees*. He showed that the decision problem Higher-Order Matching is reducible to the ADTA non-emptiness problem,

and asked if the latter is decidable. The second contribution of this paper is a characterisation (Theorem 15) of the problem v-REACH $[\text{fPCF}^\star, \text{fPCF}]$ (for $v \in o$) in terms of ADTA acceptance and ADTA non-emptiness problems. Thanks to the preceding undecidability result, we obtain the undecidability of ADTA non-emptiness as a corollary.

Theorem 15 is proved using a characterisation of fPCF* computation by *traversals* [19], [4], [3]. A traversal over the *full computation tree* (which is a souped-up syntax tree) of a term M, $\lambda^{\mathrm{f}}(M)$, is a certain sequence of nodes of the tree; unlike a path in the tree, a traversal can "jump" all over the tree.[1] Given a closed fPCF*-term M, we construct an ADTA that simulates traversals t over $\lambda^{\mathrm{f}}(M)$ by a set of paths that correspond to the P-views of prefixes of t. The states of the simulating ADTA are based on *variable profiles* [19], which are assertions about the value bound to a variable when control (in the form of a traversal) reaches it.

Our third contribution concerns a relativised reachability problem. By restricting to contexts constructible from a finite set of variable names, we show that the corresponding solution set of a given instance of $\star$-REACH $[\text{fPCF}^\star, \text{fPCF}]$ is recognisable by an alternating tree automaton, and hence regular. Thus the relativised problem is decidable. As a corollary, $\star$-REACH $[\text{fPCF}^\star, \text{fPCF}]$ is decidable at order 3.

Related work

The aim of *Control Flow Analysis* (CFA) is to approximate the flow of control within a program phrase in the course of a computation (see e.g. Midtgaard's survey [16] and the book by Nielson et al. [18]). In a functional computation, control flow is determined by a sequence of function calls (possibly unknown at compile time); thus CFA amounts to approximating the values that may be substituted for bound variables during the computation. Since these values are (denoted by) pieces of syntax, CFA reduces to an algorithm that assigns *closures* (subterms of the examined term paired with substitutions for free variables) to bound variables. Reachability analysis and CFA are clearly related: for example, the former can aid the latter because unreachable parts of the term can be safely excluded from the range of closure assignment. There are however important differences: on one hand, CFA algorithms are *approximation algorithms* designed to address a more general problem; on the other, because CFA considers terms in isolation of its possible (program) contexts, the corresponding notion of reachability essentially amounts to reachability in the reduction graph.

Also relevant are (mainly type-theoretic [2], [9]) methods to detect *useless code* (which subsumes dead code). A subterm of a term is *useless* if it does not contribute to the evaluation. State-of-the-art algorithms employ only static information, without predicting the values of bound variables or analysing control flow. Consequently, these algorithms offer even coarser approximations than CFA.

Based on the fully abstract game semantics, traversals are a (particularly accurate) model of the flow of control within a term; they can therefore be viewed as a CFA method. In fact, Hankin and Malacaria [15], [14] proposed a game-semantical approach to CFA. Their work utilised a kind of traversals over what they call *flowcharts*, a construction similar (but not identical) to Blum-Ong [4], [19].

Outline

The rest of the paper is organised as follows. Section II introduces the decision problem template θ-REACH $[\mathcal{L}_1, \mathcal{L}_2]$ and establishes the undecidability results. Section III introduces Σ-binding trees and traversals. Section IV characterises v-REACH $[\text{fPCF}^\star, \text{fPCF}]$, proves the undecidability of the ADTA non-emptiness problem, and the decidability of a relativised reachability problem.

II. REACHABILITY IN HIGHER-ORDER COMPUTATION

Consider (boolean) PCF [20], which is the simply-typed lambda calculus generated from the base type of booleans $o := \{\mathsf{t}, \mathsf{f}\}$, augmented with a definition-by-cases construct (or conditional) and a fixpoint operator at every type:

$$\begin{array}{ll} \textit{Types} & A, B ::= o \mid A \to B \\ \textit{Terms} & M, N ::= \mathsf{t} \mid \mathsf{f} \mid \mathsf{if} \mid x \mid \lambda x.M \mid MN \mid \mathbf{Y}_A \end{array}$$

Each type A can be written in the form $A_1 \to \cdots \to A_n \to o$ (by convention, arrows associate to the right) which we abbreviate to $(A_1, \cdots, A_n, o)$. The number n, denoted $ar(A)$, is called the ***arity*** of A. The ***order*** of A is given recursively by $ord(A) := \max\{ord(A_1) + 1, \cdots, ord(A_n) + 1, 0\}$; thus $ord(o) = 0$. *Reduction* in PCF is defined by means of a small-step reduction relation, with redexes as follows

$$\begin{array}{ll} (\beta) & (\lambda x^A.M)\, N \to M[N/x] \\ (\mathbf{Y}) & \mathbf{Y}_A\, M \to M\,(\mathbf{Y}_A\, M) \end{array} \qquad (\mathsf{if}) \quad \begin{cases} \mathsf{if}\,\mathsf{t} \to \lambda x^o y^o.x \\ \mathsf{if}\,\mathsf{f} \to \lambda x^o y^o.y \end{cases}$$

and evaluation contexts defined by $E ::= [\text{-}] \mid E\,M \mid \mathsf{if}\,E$. Because of recursion, PCF is not normalising. We say that M ***observationally approximates*** N, written $M \lesssim N$, just if for every program context $C[\text{-}]$ and $v \in \{\mathsf{t}, \mathsf{f}\}$, if $C[M] \twoheadrightarrow v$ then $C[N] \twoheadrightarrow v$. This yields a notion of ***observational equivalence***: $M \cong N$ if $M \lesssim N$ and $N \lesssim M$.

Given a PCF term M, we can think of a program point as a pair L^α, where L is a subterm of M, and α, which is a path in the syntax tree of M, indicates an occurrence of L in M. Let $C[\text{-}]$ be a program context for M. As the computation of $C[M]$ unfolds, L^α may be eliminated, copied or modified (by substitution) as a result of the rewrite rules — the terms into which L^α may evolve in this way are called *residuals*. We can express Contextual Reachability as follows.

[1] Intuitively, and using the language of game semantics, a traversal over $\lambda^{\mathrm{f}}(M)$ is a representation of an interaction sequence which is obtained by hereditarily *uncovering* (in the sense of Hyland and Ong [8, p. 341]) a play in the strategy-denotation of M.

CONTEXTUAL REACHABILITY: Given a PCF term M and a program point given by an occurrence α of a subterm L of M, is there a program context $C[\text{-}]$ for M such that $C[M] \twoheadrightarrow E[L^{\alpha}_{\sigma}]$, for some evaluation context $E[\text{-}]$ and substitution σ?

Because the reduction is leftmost and call-by-name, it suffices to consider only *applicative* contexts (thanks to a Context Lemma [17]). We can reformulate the decision problem as follows.

REACHABILITY: Given a closed PCF term $M : (A_1, \cdots, A_n, o)$ and an occurrence α of a subterm L of M, are there closed PCF terms $N_1, \ldots, N_n$ such that $M\overline{N} \twoheadrightarrow E[L^{\alpha}_{\sigma}]$, for some evaluation context $E[\text{-}]$ and substitution σ?

For example take the term

$$M \;:=\; \lambda\varphi\, x\,.\,\mathsf{if}\,(\varphi\, x)\,(\mathsf{if}\,(\varphi\,(\varphi\, x))\,P\,L)\,Q$$

of type $(o \to o) \to o \to o$. Then L is "reachable" using the test-terms $\mathsf{neg} : o \to o$ (negation function) and $\mathsf{f} : o$.

$$\begin{array}{rcl} M\,\mathsf{neg}\,\mathsf{f} & \twoheadrightarrow & \mathsf{if}\,\mathsf{t}\,(\mathsf{if}\,(\mathsf{neg}\,(\mathsf{neg}\,\mathsf{f}))\,P_\sigma\,L_\sigma)\,Q_\sigma \\ & \to & \mathsf{if}\,(\mathsf{neg}\,(\mathsf{neg}\,\mathsf{f}))\,P_\sigma\,L_\sigma \;\twoheadrightarrow\; L_\sigma \end{array}$$

Is REACHABILITY decidable? In the course of our investigation, we shall consider a related language PCF* and several of its finitary sublanguages, and ask if reachability is decidable in each case.

PCF-with-error: PCF*

By PCF* we mean PCF terms generated from the base type $o := \{\mathsf{t}, \mathsf{f}, \star\}$. The distinguished constant $\star$ is an error constant in the sense of Cartwright et al. [5]. The redexes of PCF* are those of PCF and

$$(\mathsf{if}\,\star) \quad \mathsf{if}\,\star \to \lambda x^o y^o.\star$$

while evaluation contexts remain the same. Thus, as soon as $\star$ is encountered, it is propagated to the outer-most level (in particular, we have $\mathsf{if}\,\star\,M\,N \;\twoheadrightarrow\; \star$). The notion of observational approximation is now given by: $M \lesssim N$ just if for every program context $C[\text{-}]$ and every $v \in \{\mathsf{t}, \mathsf{f}, \star\}$, if $C[M] \twoheadrightarrow v$ then $C[N] \twoheadrightarrow v$. Note that $\star \not\lesssim \mathsf{t}, \mathsf{f}$.

REACHABILITY can now be given the following equivalent but simpler formulation.

$\star$-REACHABILITY: Given a closed PCF*-term $M : (A_1, \cdots, A_n, o)$ that has exactly one occurrence of $\star$, are there closed PCF terms $N_1, \cdots, N_n$ such that $M\overline{N} \twoheadrightarrow \star$?

Lemma 1: The decision problems REACHABILITY *and* $\star$-REACHABILITY *are (polynomially) reducible to each other.*

For ease of comparison between problems, we introduce the following ***decision problem template*** with parameters θ, $\mathcal{L}_1$ (*principal term*) and $\mathcal{L}_2$ (*test-terms*), where $\mathcal{L}_1$ and $\mathcal{L}_2$ are sublanguages of PCF*, and θ is a base-type value of $\mathcal{L}_1$.

θ-REACH $[\mathcal{L}_1, \mathcal{L}_2]$: Given a closed $\mathcal{L}_1$-term $M : (A_1, \cdots, A_n, o)$, are there closed $\mathcal{L}_2$-terms $N_1, \cdots, N_n$ such that $M\overline{N} \twoheadrightarrow \theta$?

Using the template, $\star$-REACHABILITY is the problem $\star$-REACH $[\text{PCF}^{1\star}, \text{PCF}]$, where PCF^{1*} is the set of PCF*-terms that have exactly one occurrence of $\star$.

Finitary sublanguages: fPCF *and* fPCF*

Let us now consider the respective *finitary* (i.e. recursion-free) and hence (strongly) normalizing fragments of PCF and PCF*, written fPCF and fPCF*. It is straightforward to see that there is no loss of generality in restricting to finitary test-terms:

Lemma 2: The decision problems $\star$-REACH $[\text{PCF}^{1\star}, \text{PCF}]$ *and* $\star$-REACH $[\text{PCF}^{1\star}, \text{fPCF}]$ *are equivalent.*

Henceforth we focus on finitary languages. Further, we may WLOG restrict our attention to principal- and test-terms that are β-normal forms (β-NF). A first result is the following.

Lemma 3: The decision problems $\star$-REACH $[\text{fPCF}^{1\star}, \text{fPCF}]$ *and* $\star$-REACH $[\text{fPCF}^\star, \text{fPCF}]$ *are (polynomially) reducible to each other.*

Note that $\star$-REACH $[\text{fPCF}^{1\star}, \text{fPCF}]$ is non-trivial. To check if a given fPCF^{1*}-term $M : (A_1, \cdots, A_n, o)$ is a yes-instance, it suffices to evaluate $M\overline{N}$ for a representative N_i of each observational equivalence class of each type A_i. Unfortunately these classes are not effectively presentable: the equivalence is taken in fPCF* where Loader's result [13] applies (see below). In contrast, v-REACH $[\text{fPCF}, \text{fPCF}]$ ($v \in \{\mathsf{t}, \mathsf{f}\}$) is decidable, because the equivalence classes are given by elements of the appropriate type in the hierarchy of higher-order functions on a boolean base type [23].

Finitary PCF-with-bottom: fPCF$_\perp$

Next we introduce *finitary PCF-with-bottom* (as considered by Loader [13]), written fPCF$_\perp$, which is syntactically just fPCF* with $\star$ replaced by $\perp$; the reduction rules are the same as in fPCF*. As a result, the two languages have the same notion of observational equivalence[2] (which is undecidable [13]), which implies the following equivalence.

Lemma 4: For every $v \in \{\mathsf{t}, \mathsf{f}, \star\}$, *setting* $v' := \perp$ *if* $v = \star$ *and* $v' := v$ *otherwise,* v-REACH $[\text{fPCF}^\star, \text{fPCF}]$ *and* v'-REACH $[\text{fPCF}_\perp, \text{fPCF}]$ *are (polynomially) reducible to each other.*

[2] in the sense that $M \cong N$ in fPCF* iff $M[\perp/\star] \cong N[\perp/\star]$ in fPCF$_\perp$. Note that because reduction to $\perp$ means *divergence* rather than error, the two languages have distinct notions of observational approximation.

Finitary PCF-with-error-and-bottom: $\text{fPCF}^{\star}_{\perp}$

We do not know if the reachability problems of Lemma 4 are decidable. On the other hand, taking $\text{fPCF}^{\star}_{\perp}$ to be the (finitary) language generated from base type $o := \{\mathsf{t}, \mathsf{f}, \perp, \star\}$, its $\star$-REACHABILITY problem (i.e. $\star$-REACH $[\text{fPCF}^{\star}_{\perp}, \text{fPCF}]$) is undecidable. As before, we write $\text{fPCF}^{1\star}_{\perp}$ for the set of $\text{fPCF}^{\star}_{\perp}$ terms in which $\star$ occurs exactly once.

Lemma 5: If $\star$-REACH $[\text{fPCF}^{1\star}_{\perp}, \text{fPCF}]$ *is decidable, then so is* t-REACH $[\text{fPCF}_{\perp}, \text{fPCF}]$.

Proof: $M \in \mathsf{t}$-REACH $[\text{fPCF}_{\perp}, \text{fPCF}]$ iff $\lambda\overline{x}.\,\mathsf{if}\,(M\overline{x})\star\perp \in \star$-REACH $[\text{fPCF}^{1\star}_{\perp}, \text{fPCF}]$, for every $\text{fPCF}_{\perp}$-term M. ■

Lemma 6: If t-REACH $[\text{fPCF}_{\perp}, \text{fPCF}]$ *is decidable, then so is the problem: Given a system of* $\text{fPCF}_{\perp}$*-equations*

$$\begin{cases} X\,a_1^1\,\ldots\,a_n^1 \;\cong\; b^1 \\ X\,a_1^2\,\ldots\,a_n^2 \;\cong\; b^2 \\ \qquad\cdots \\ X\,a_1^m\,\ldots\,a_n^m \;\cong\; b^m \end{cases} \tag{1}$$

where each a_i^j *is a term of type* A_i *and each* $b^j \in \{\mathsf{t}, \mathsf{f}\}$*, is there a solution (in* $\text{fPCF}_{\perp}$*) for* $X : (A_1, \ldots, A_n, o)$*?*

Proof: Note that t-REACH $[\text{fPCF}_{\perp}, \text{fPCF}_{\perp}]$ and t-REACH $[\text{fPCF}_{\perp}, \text{fPCF}]$ are equivalent. Given such a system of equations we can construct a term $G : (o, \ldots, o, o)$ such that, for each $c_i : o$,

$$G\,c_1 \cdots c_n \quad\cong\quad \begin{cases} \mathsf{t} & \text{if } c_i \cong b^i \text{ for each } i \\ \perp & \text{otherwise} \end{cases}$$

in $\text{fPCF}_{\perp}$. Take $M : ((A_1, \ldots, A_n, o), o)$ to be

$$\lambda X.\mathsf{if}\;(G\,(X\,a_1^1\,\ldots\,a_n^1)\ldots(X\,a_1^m\,\ldots\,a_n^m))\;\mathsf{t}\;\perp.$$

Then, the system of equations has a solution in $\text{fPCF}_{\perp}$ iff $M \in \mathsf{t}$-REACH $[\text{fPCF}_{\perp}, \text{fPCF}_{\perp}]$. ■

Corollary 7: The following problems are undecidable.

(i) t-REACH $[\text{fPCF}_{\perp}, \text{fPCF}]$, t-REACH $[\text{fPCF}^{\star}, \text{fPCF}]$
(ii) t-REACH $[\text{fPCF}_{\perp}, \text{fPCF}_{\perp}]$
(iii) $\star$-REACH $[\text{fPCF}^{1\star}_{\perp}, \text{fPCF}]$, $\star$-REACH $[\text{fPCF}^{\star}_{\perp}, \text{fPCF}]$
(iv) $\star$-REACH $[\text{PCF}^{1\star}, \text{PCF}]$
(v) $\star$-REACH $[\text{PCF}^{\star}, \text{PCF}]$
(vi) REACHABILITY

Proof: (i) Loader [13] has shown that solvability of the system of equations (1) (as defined in Lemma 6) is undecidable. (ii) is equivalent to (i), (iii) follows from (i), and (v) follows from (iv). (iv) is undecidable because of Lemma 2 and the fact that $\text{fPCF}^{1\star}_{\perp}$ is a sublanguage of $\text{PCF}^{1\star}$; (vi) then follows from Lemma 1. ■

Solvability of the system of equations (1) is undecidable at order 3 [13]. Hence problems (i-vi) above are undecidable at order 4 onwards. For (i, ii), this is optimal because $\text{fPCF}_{\perp}$-test-terms of order 2 can be effectively generated [21]. Corollary 19 shows that it is optimal for (iii) too.

We have seen that the following problems are polynomially reducible to each other. Are they decidable?

(i) $\star$-REACH $[\text{fPCF}^{\star}, \text{fPCF}]$
(ii) $\perp$-REACH $[\text{fPCF}_{\perp}, \text{fPCF}]$

Much of the rest of the paper is concerned with this question.

III. Σ-BINDING TREES AND TRAVERSALS

Assume Σ is a ranked alphabet (i.e. each symbol $s \in \Sigma$ has an arity $ar(s) \geq 0$) that is partitioned into $\Sigma_{\lambda}, \Sigma_{\text{var}}$ and Σ_{cst}, where Σ_{λ} consists of *binders* of arity 1, Σ_{var} consists of *variables*, and Σ_{cst} consists of *constants*. A Σ***-tree*** is a finite Σ-(node-)labelled tree that satisfies:

> *Bipartition*: If node n is Σ_{λ}-labelled then $n1$ is labelled in $\Sigma_{\text{var}} \cup \Sigma_{\text{cst}}$; and if node n is labelled in $\Sigma_{\text{var}} \cup \Sigma_{\text{cst}}$ and ni is a successor then it is Σ_{λ}-labelled.

The ***long transform*** of a $\text{fPCF}^{\star}$-term is obtained by: (i) hereditarily η-expanding every subterm (even if it is of base type so that $e : o$ expands to $\lambda.e : o$, a term with a "dummy" lambda) provided it does not occur as the first argument of the application operator; then (ii) inserting *long-apply symbols* $@^{A\to A}$ i.e. replacing every base-type subterm of the shape $(\lambda\overline{x}.P)\,Q_1 \cdots Q_n$ where $n \geq 1$ by $@\,(\lambda\overline{x}.P)\,Q_1 \cdots Q_n$.[3] E.g. (to avoid notational clutter, we shall often omit type superscripts of variables and constants)

$$(\lambda\varphi^{(o,o)}.\varphi)\,(\lambda x^o.x)\,\mathsf{t} \;\mapsto\; \lambda.@^{A\to A}(\lambda\varphi\,y.\varphi(\lambda.y))\,(\lambda x.x)\,(\lambda.\mathsf{t})$$

where $A = ((o,o),o,o)$.

The ***computation tree*** of a $\text{fPCF}^{\star}$-term M, written $\lambda(M)$, is its long transform viewed as a Σ-tree with (typed, partitioned) alphabet $\Sigma = \Sigma_{\lambda} + \Sigma_{\text{var}} + \Sigma_{\text{cst}}$ given as follows. Each symbol s^A has arity $ar(A)$.

- Σ_{cst} is a finite subset of $\{\mathsf{if}^{(o,o,o,o)}, \mathsf{t}^o, \mathsf{f}^o, \star^o\} \cup \{@^{A\to A} : ar(A) \geq 1\}$.
- Σ_{var} is a finite set of (typed) variables.
- Σ_{λ} is a finite subset of $\{\lambda x_1 \cdots x_n : n \geq 0,\, x_i^{A_i} \in \Sigma_{\text{var}}\}$ where each $\lambda x_1^{A_1} \cdots x_n^{A_n}$ has type $(A_1, \cdots, A_n, o)$.

We call such a Σ a $\text{fPCF}^{\star}$***-alphabet***.

Example 3.1: Consider the $\text{fPCF}^{\star}$-term M:

$$(\lambda\Phi\,x^o.\Phi\,(\lambda y^o.\mathsf{if}\;y\,(\Phi(\lambda z^o.z)\star)\,x)\,\mathsf{t})\;(\lambda\varphi x^o.\varphi x)\;\mathsf{t} \;:\; o$$

Its computation tree $\lambda(M)$ is shown in Figure 1 (ignore dotted arrows for now). Observe that nodes on levels 0, 2, 4, etc. are labelled with "lambdas", and those on levels 1, 3, 5, etc. are labelled with non-lambda symbols — this is just the bipartition condition.

A Σ***-binding tree***, in the sense of Stirling [22], is a Σ-tree which is equipped with a *binding* relation between nodes such that, for every Σ_{var}-labelled node n, there is a unique ancestor node b of n that is Σ_{λ}-labelled and *binds* n, written $b \curvearrowleft n$. An important example of Σ-binding tree is the computation tree of a (closed) $\text{fPCF}^{\star}$-term, where Σ is the underlying $\text{fPCF}^{\star}$-alphabet: the binding relation $b \curvearrowleft n$ is just the standard binding relation between nodes n labelled with a variable x_i and nodes b labelled with a λ-binder of the form $\lambda x_1 \cdots x_k$, with $1 \leq i \leq k$.[4] For example, in Figure 1, the binding relation is shown as dotted arrows.

[3] Note that $A \to A = (A, A_1, \cdots, A_n, o)$ for $A = (A_1, \cdots, A_n, o)$.

[4] The binding relation "$b \curvearrowleft n$" corresponds to the presence of a *justification pointer*, in a game-semantic reading of $\text{fPCF}^{\star}$ terms [8], [4], [3].

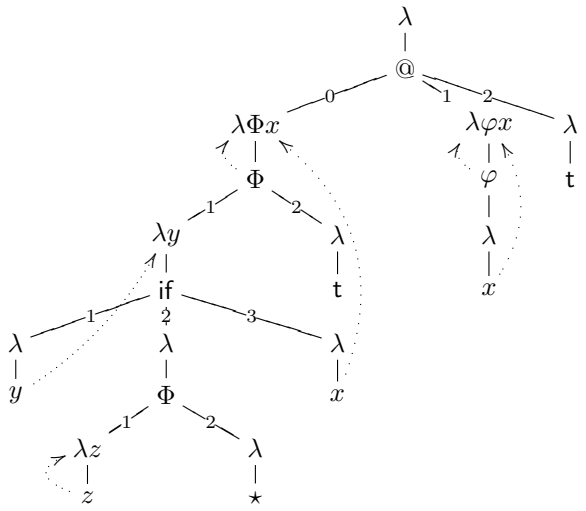

Figure 1. A computation tree. For technical convenience, the leftmost child of a @-labelled node is its 0-child; the leftmost child of every other node is its 1-child.

Remark 3.2: It is not possible, in general, to use a fixed, finite set of variable-names to construct an infinite family of simply-typed lambda terms of the same type. Consider, for example, the *monster type* $((((o,o),o),o),o,o)$, which has the following family of terms,

$$M_n := \lambda\varphi\, x.\varphi(\lambda z_1.(\varphi(\lambda z_2.(\cdots \varphi(\lambda z_n.z_n(\cdots z_2(z_1\, x)))))))).$$

for $n \geq 1$ [22]. The key advantage of representing computation trees as Σ-binding trees is that we can "economise" on variable names (see Lemma 8). For example, the preceding family of terms can be represented as Σ-binding trees, each using only two variable names and the following bindings (indicated by dotted arrows).

$$\lambda\overset{\curvearrowleft}{\varphi}\, x.\, \varphi(\lambda\overset{\curvearrowleft}{z}.\, (\cdots \varphi(\lambda\overset{\curvearrowleft}{z}.\, z(\cdots(z\, x))))) \qquad (2)$$

Let us now assume that each subtype B of a type A is implicitly *labelled* with its occurrence α in A, written $B^\alpha \leq A$, so that different occurrences of the same type are distinguished[5]; we use $<$ for the *proper* subtype relation. To avoid notational clutter, we shall omit the occurrence of a subtype whenever what we mean is clear from the context. The subtypes of a type $A = (A_1, \cdots, A_n, o)$ are partitioned into two groups[6], $O(A)$ and $P(A)$, recursively by:

$$O(A) := \{A\} \cup \bigcup_{i=1}^{n} P(A_i)\,, \quad P(A) := \bigcup_{i=1}^{n} O(A_i)\,.$$

Lemma 8: For every type A, at most $|P(A)|$ variable-names are needed to represent every *closed* fPCF**-term $M : A$ in β-NF as a binding tree.*

[5]For example, the monster type $((((o,o),o),o),o,o)$ has five proper (labelled) subtypes, namely o^{1111}, $(o,o)^{111}$, $((o,o),o)^{11}$, $(((o,o),o),o)^1$ and o^2.

[6]They correspond to O- and P-moves in the game semantics of A.

By a ***variable assignment*** Ξ of type A, written $\Xi : A$, we mean an injective map Ξ from (labelled) proper subtypes B of A to finite sets of variable names (written $\Xi : B \mapsto \Xi^B$), such that if B_1 and B_2 are distinct subtypes, then $\Xi^{B_1} \cap \Xi^{B_2} = \emptyset$. We may write ξ^B to denote an element of Ξ^B. The ***associated*** fPCF****-alphabet*** Σ_Ξ has the following components,

$$\begin{array}{rcl} \Sigma_\lambda & := & \{\lambda\xi_1^{B_1}\cdots\xi_m^{B_m} : (B_1,\cdots,B_m,o) \leq A\} \\ \Sigma_{\mathrm{var}} & := & \bigcup_{B<A} \Xi^B \\ \Sigma_{\mathrm{cst}} & := & \mathit{Val} \cup \{\mathsf{if}, @^{A\to A}\} \end{array}$$

where $\mathit{Val} := \{\mathsf{t}, \mathsf{f}, \star\}$.[7] A variable assignment Ξ is said to be ***singleton*** just if Ξ^B is a singleton set for every B (and if so, we write $\Xi^B = \{\xi^B\}$).

We say that a Σ_Ξ-binding tree is ***well-typed*** just if: (i) whenever a node has a Σ_{var}- or Σ_{cst}-label of type $B = (B_1, \cdots, B_m, o)$, then it has m children, and its i-th child has a Σ_λ-label of type B_i; (ii) if a node b has a Σ_λ-label $\lambda\xi_1^{B_1}\cdots\xi_m^{B_m}$, and $b \curvearrowleft n$, then n has Σ_{var}-label $\xi_i^{B_i}$.

Lemma 9: Let $\Xi : A = (A_1, \cdots, A_n, o)$ be a singleton variable assignment. Every closed fPCF**-term $M : B \leq A$ in β-NF is* representable *as a well-typed Σ_Ξ-binding tree, which is written $\lambda_\Xi(M)$.* [8] *In case $M : A$, the same holds for $M\overline{N} : o$ where each $N_i : A_i$ is a closed β-NF.*

Lemma 10: Let $\Xi : A = (A_1, \cdots, A_n, o)$ be a variable assignment. Every well-typed *Σ_Ξ-binding tree* represents *some closed* fPCF**-term.*

Traversals over a full Σ-binding tree

Traversals [19], [4], [3] are a representation of higher-order computation based on fully abstract game semantics [8]. Let M be a fPCF*-term. Formally a traversal over the *full computation tree* of M (which is an extension of $\lambda(M)$ with value nodes), $\lambda^{\mathrm{f}}(M)$, is a certain sequence of nodes of the tree that starts from the root). However, unlike a path in the tree, a traversal can "jump" all over the tree. Intuitively traversals capture the flow of control in $\lambda^{\mathrm{f}}(M)$ from the perspective of game semantics. Here we define traversals over a Σ-binding tree[9] T that represents M.

We define ***enabling***, a family of binary relations $\vdash_i$ between nodes of the tree T where $0 \leq i \leq m$ (where m is the largest arity of the types of the subterms of M) by the following rules.

(e1) Every lambda node β, which is the i-child[10] of its parent node α, is *i-enabled* by α.

[7]Because we assume that the principal term is in $\beta\eta$-LNF, we need only one long-apply symbol, which is at the topmost level, of type $A \to A$.

[8]e.g. the binding tree of (2) in Remark 3.2.

[9]This subsumes the two cases we shall consider in the sequel: (i) $T = \lambda_\Xi(M)$ where Ξ is a singleton variable assignment (corresponding to Theorem 15); (ii) $T = \lambda(M)$ where M is a $\beta\eta$-LNF that is consistent with a variable assignment Ξ (corresponding to Theorem 17).

[10]Recall the ordering convention in computation trees (Figure 1).

(e2) If the node β, which is labelled with variable ξ_i, is bound by node α which is labelled with $\lambda\xi_1 \cdots \xi_n$, then we say that β is *i-enabled* by α.

The ***full tree*** of T, written T^{f}, is obtained from T by adding, for each $v \in Val$, a v-labelled leaf[11] as a child of every node of the tree, except those nodes labelled by an element of Val and their respective parent nodes. We extend the enabling relation of T to T^{f} by an additional rule:

(e3) For each leaf node n that is labelled by a value $v \in Val$, we say that n is *v-enabled* by its parent node.

We say that node β is ***enabled*** by α just if β is x-enabled by α, for some (necessarily unique) $x \in \{0, \cdots, m\} \cup Val$.

A node of T^{f} is ***initial*** if it is not enabled by any node. It follows that the initial nodes of a full computation tree are the root node and nodes that are labelled with either a long-apply symbol or if. An ***O-node*** is either a lambda node, or a leaf whose parent is not a lambda node. A ***P-node*** is a node that is not labelled with a lambda, or a leaf whose parent is a lambda node. Thus, nodes of a full computation tree that occur on levels 0, 2, 4, etc. are O-nodes, and those that occur on levels 1, 3, 5, etc. are P-nodes. A node of a full computation tree is an ***answer node*** if it is labelled with an element of Val; otherwise it is a ***question node***.

A ***justified sequence*** over T^{f} is an O/P-alternating sequence of nodes that satisfies the *pointer condition*: Every non-initial node that occurs in it has a pointer to some earlier occurrence of the node (in T^{f}) which enables it. *Notation*: $\cdots n_0 \overset{j}{\cdots} n$ means that n points to n_0 and n is j-enabled by n_0. We say that n is *j-justified* by n_0 in the justified sequence. Note that henceforth by *program* we mean a closed term of base type.

Definition 3.3: Let P be a fPCF* program, and T be a Σ-binding tree representing P. ***Traversals*** over the full tree T^{f} are justified sequences of nodes defined by induction over the following rules. We let t range over traversals, n over nodes labelled by non-Σ_λ symbols, and u and v over Val. For ease of reading, we refer to nodes of T^{f} by their labels.

1) (*Root*): The singleton sequence, comprising the root node ε, is a traversal.
2) (*App*): If $t \cdot @$ is a traversal, so is $t \cdot @ \overset{0}{\cdot} \lambda\overline{\xi}$.
3) (*Lam*): If $t \cdot \lambda\overline{\xi}$ is a traversal and n is the 1-child of node $\lambda\overline{\xi}$ in T^{f}, then so is $t \cdot \lambda\overline{\xi} \cdot n$.
4) (*Var*): If $t \cdot n \cdot \lambda\overline{\xi} \overset{i}{\cdots} \xi$ is a traversal where n is *not* labelled by if, then so is $t \cdot n \overset{i}{\cdot} \lambda\overline{\xi} \overset{i}{\cdots} \xi \cdot \lambda\overline{\eta}$.
5) (*O-Val*): If $t \cdot \lambda\overline{\xi} \cdot n \cdots v$ is a traversal, so is $t \cdot \lambda\overline{\xi} \cdot n \cdots v \cdot v$.
6) (*P-Val*): If $t \cdot n \cdot \lambda\overline{\xi} \cdots v$ is a traversal where node n is labelled by $@$ or a variable, then so is $t \cdot n \cdot \lambda\overline{\xi} \cdots v \cdot v$.
7) (*If*): If $t \cdot \mathsf{if}$ is a traversal, so is $t \cdot \mathsf{if} \overset{1}{\cdot} \lambda$.
8) (*If*-t): If $t \cdot \mathsf{if} \overset{1}{\cdot} \lambda \cdots \mathsf{t}$ is a traversal, so is $t \cdot \mathsf{if} \overset{2}{\cdot} \lambda \overset{1}{\cdots} \mathsf{t} \cdot \lambda$.
9) (*If*-f): If $t \cdot \mathsf{if} \overset{1}{\cdot} \lambda \cdots \mathsf{f}$ is a traversal, so is $t \cdot \mathsf{if} \overset{3}{\cdot} \lambda \overset{1}{\cdots} \mathsf{f} \cdot \lambda$.
10) (*If*-$\star$): If $t \cdot \mathsf{if} \overset{1}{\cdot} \lambda \cdots \star$ is a traversal, so is $t \cdot \mathsf{if} \overset{1}{\cdot} \lambda \cdots \star \cdot \star$.
11) (*If-CC*): If $t \cdot \mathsf{if} \overset{1}{\cdot} \lambda \cdots v \overset{i}{\cdot} \lambda \cdots u$ is a traversal, with $i \in \{2, 3\}$, then so is $t \cdot \mathsf{if} \overset{1}{\cdot} \lambda \cdots v \overset{i}{\cdot} \lambda \cdots u \cdot u$.

It follows that the way a traversal can grow is deterministic. A traversal over T^{f} is said to be *v-**complete*** just if every question node that occurs in it justifies some answer node; further, the opening question (v-)justifies a v-labelled node.

Theorem 11 (Correspondence): Let P be a fPCF**-program, T a binding tree that represents P, and $v \in Val$. We have $P \twoheadrightarrow v$ iff there is a v-complete traversal over T^{f}.*

Proof: Using the language of game semantics, a traversal over T^{f} is a representation of an interaction sequence which is obtained by hereditarily *uncovering* (in the sense of Hyland and Ong [8, p. 341]) a play in the strategy-denotation of M. See preprint [4] or Blum's thesis [3] for a proof. ■

Example 3.4: Let Ξ be the variable assignment of Example 4.2. Consider the base-type term MN where:

$$M := \lambda\Phi.\Phi\,(\lambda\varphi.\varphi\,(\lambda x.x)\,(\lambda.\Phi\,(\lambda\varphi.\mathsf{t})\,(\lambda.\varphi\,(\lambda x.\star)\,(\lambda.\mathsf{t}))))\,(\lambda.\mathsf{f})$$
$$N := \lambda\psi y.\mathsf{if}\,(\lambda.\psi\,(\lambda\alpha\,z.\alpha\,(\lambda.z)))\,(\lambda.y)\,(\lambda.y).$$

In Figure 2, we display a $\star$-complete traversal over $\lambda^{\mathrm{f}}(MN)$, which is the justified sequence $1 \cdot 2 \cdot 3 \cdots 59 \cdot 60$, where the numbers are node-names indicated as superscripts.

We close this section by recalling the notion of *view* (of a justified sequence) and the condition of *Visibility*, which were introduced in game semantics [8]. Intuitively the *P-view* of a justified sequence is a certain subsequence consisting of moves which the player P considers relevant for determining his next move in the play. Formally the ***P-view***, $\ulcorner t \urcorner$, of a justified sequence t is a subsequence defined by recursion as follows, where we let o range over O-nodes and p over P-nodes.[12]

$$\begin{aligned} \ulcorner t \cdot o \urcorner &:= o \qquad \text{if node } o \text{ is initial} \\ \ulcorner t \cdot p \overset{i}{\cdots} o \urcorner &:= \ulcorner t \urcorner \cdot p \overset{i}{\cdot} o \\ \ulcorner t \cdot p \urcorner &:= \ulcorner t \urcorner \cdot p \end{aligned}$$

A justified sequence t satisfies ***P-visibility*** just in case every non-initial P-node that occurs in t points to some (necessarily O-) node appearing in the P-view at that point.

[11] By a *leaf* of a tree, we mean a terminal node.

[12] In the third clause, suppose the P-node p points to some node-occurrence l (say) in t; if l appears in $\ulcorner t \urcorner$, then p in $\ulcorner t \urcorner \cdot p$ is defined to point to l; otherwise p has no pointer.

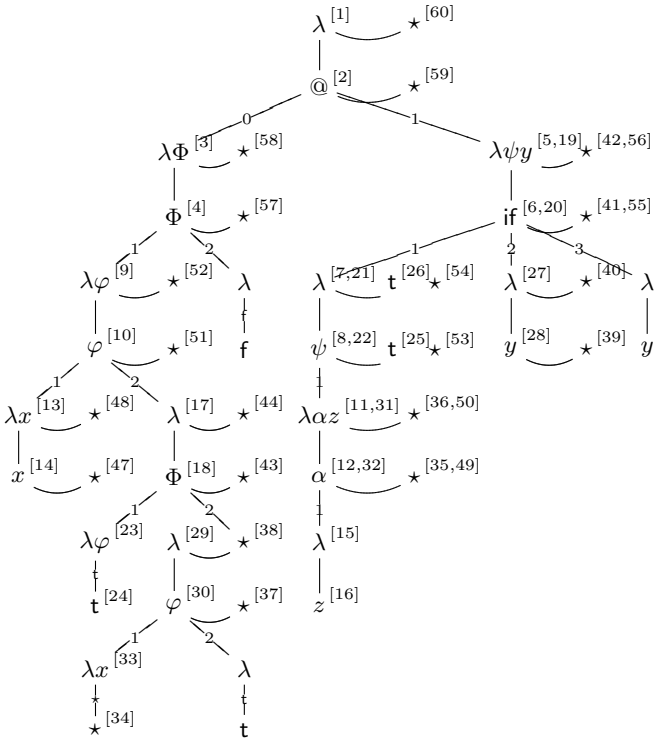

Figure 2. A $\star$-complete traversal over the full computation tree of MN as defined in Example 3.4. Note that only those value leaves that participate in the traversal are depicted.

IV. Alternating dependency Σ-tree automata and characterisation of reachability

Definition 4.1 (Stirling [22]): An ***alternating dependency Σ-tree automaton*** (ADTA) is a quadruple $\mathsf{A} = \langle\, Q, \Sigma, q_0, \Delta \,\rangle$ where Q is a finite set of states, Σ is a finite alphabet, $q_0 \in Q$ is the initial state, and Δ is a finite set of transition rules each of which has one of the following forms.

(i) $q\,s \Rightarrow (Q_1, \cdots, Q_k)$ where $s \in \Sigma_{\rm var} \cup \Sigma_{\rm cst}$, $ar(s) = k$, $q \in Q$ and $Q_1, \cdots, Q_k \subseteq Q$.
(ii) $q\,s \Rightarrow q'\,s'$ where $s \in \Sigma_\lambda$, $s' \in \Sigma_{\rm cst}$, and $q, q' \in Q$
(iii) $(q', q)\,s \Rightarrow q_1\,x$ where $s \in \Sigma_\lambda, x \in \Sigma_{\rm var}$ and $q', q, q_1 \in Q$.

A ***run-tree*** of an ADTA $\langle\, Q, \Sigma, q_0, \Delta \,\rangle$ on a Σ-binding tree t is a $(\Sigma \times Q)$-labelled (unranked) tree whose nodes are pairs of the form (n, α) where n is a node of t and $\alpha \in Q^+$ is a non-empty sequence of states, labelled (s, q) if n is labelled s in t and q is the last element of the sequence α, which is defined top-down with root node (ε, q_0) where ε is the root node of t. Consider a node (n, α) of a partial run-tree which does not have successors.

(i) If $s \in \Sigma_{\rm var} \cup \Sigma_{\rm cst}$ and $q\,s \Rightarrow (Q_1, \cdots, Q_k)$ is a Δ-rule, then the successors of (n, α) are $(ni, \alpha\,q')$, for each $q' \in Q_i$ and each $1 \le i \le k$.
(ii) If $s \in \Sigma_\lambda$, $n1$ is labelled $s' \in \Sigma_{\rm cst}$ and $q\,s \Rightarrow q'\,s'$ is a Δ-rule, then $(n1, \alpha\,q')$ is the successor of (n, α).
(iii) If $s \in \Sigma_\lambda$, $n1$ is labelled $x \in \Sigma_{\rm var}$, $m \curvearrowleft n1$ in t, $(m, \alpha'\,q')$ equals or is an ancestor of (n, α), and $(q', q)\,s \Rightarrow q_1\,x$ is a Δ-rule, then $(n1, \alpha\,q_1)$ is the successor of (n, α).

A run-tree of A on t is *accepting* if whenever $(n, \alpha q)$ is a leaf labelled (s, q) of the run-tree, then either s has arity 0 or the rule $q\,s \Rightarrow (\emptyset, \cdots, \emptyset)$ is in Δ. The ADTA A ***accepts*** a Σ-binding tree t if there is an accepting run-tree of A on t.

Lemma 12 (Stirling [22]): ADTA are closed under union and intersection.

Given a Σ-binding tree T that represents a closed fPCF*-term in β-NF, we construct an ADTA that simulates traversals (over the full computation tree $T^{\rm f}$) by a set of (annotated) paths in T. The states of the simulating ADTA are defined using *variable profiles* [19], which are assertions about the value bound to a variable when control — in the form of a traversal — reaches it.

Let $\Xi : A$ be a variable assignment. We define the set of ***variable profiles*** of type B w.r.t. Ξ recursively as

$$\mathbf{VP}_\Xi(B) := \{(x, v, \rho) : v \in \mathit{Val},\, x \in \Xi^B,\, \rho \subseteq \bigcup\nolimits_{i=1}^{m} \mathbf{VP}_\Xi(B_i)\}$$

where $B = (B_1, \cdots, B_m, o) < A$. Thus, in case $m = 0$, we have $\mathbf{VP}_\Xi(o) = \{(x, v, \emptyset) : x \in \Xi^o,\ v \in \mathit{Val}\}$. To save writing, we shall write base-type variable profiles as pairs (x, v). Finally, set $\mathbf{VP}_\Xi$ to be the union of sets $\mathbf{VP}_\Xi(B)$, as B ranges over proper subtypes of A. Whenever it is clear from the context, we shall omit the subscript Ξ from $\mathbf{VP}_\Xi$.

Example 4.2: Let $A = (((((o, o), o, o), o), o, o), o)$. Consider the following (singleton) variable assignment $\Xi : A$.

$$\begin{array}{c} (((((o,\ o),\ o,\ o),\ o),\ o,\ o),\ o) \\ x\ \ \alpha\ \ z\ \ \varphi\ \ \psi\ \ y\ \ \Phi \end{array}$$

Then, all of the following are variable profiles w.r.t. Ξ.

$$\theta := (\alpha, \mathsf{t}, \{(x, \mathsf{f})\}), \qquad \delta_1 := (\varphi, \mathsf{t}, \{(z, \star), \theta\}),$$
$$\eta := (\Phi, \star, \{(\psi, \star, \{\delta_1, \delta_2\})\}), \quad \delta_2 := (\varphi, \mathsf{f}, \{(\alpha, \star, \emptyset)\}).$$

Definition 4.3: Let Ξ be a singleton variable assignment of type $A = (A_1, \cdots, A_n, o)$ and Σ be Σ_Ξ or $\Sigma_\Xi \setminus \{\star\}$. The ***traversal-simulating ADTA*** $\mathsf{A}(\Sigma, q_0) := \langle\, Q, \Sigma, q_0, \Delta \,\rangle$, where $Q = \mathit{Val} \times \mathcal{P}(\mathbf{VP}_\Xi)$ and the transition relation Δ is defined as follows.

1. (*App*) $(v, \emptyset)\ @^{A\to A} \Rightarrow (\{(v, \rho)\}, Q_1, \cdots, Q_n)$ where $\rho \subseteq \bigcup_i \mathbf{VP}_\Xi(A_i)$ and $Q_i = \{(u, \rho') : (\xi^{A_i}, u, \rho') \in \rho\}$,
2. (*Var*) $(v, \rho)\ \xi^{(B_1, \cdots, B_m, o)} \Rightarrow (Q_1, \cdots, Q_m)$ where $Q_i = \{(u, \rho') : (\xi^{B_i}, u, \rho') \in \rho\}$,
3. (*Lam-V*) $((u, \rho_0), (v, \rho))\ \ \lambda\overline{\xi} \ \Rightarrow\ (v, \rho_1)\,x$ where $(x, v, \rho_1) \in \rho_0$,
4. (*Lam-C*) $(v, \rho)\ \lambda\overline{\xi} \Rightarrow (v, \emptyset)\ \$$ where $\$ \in \{\mathsf{if}, v\}$
5. (*If-*t) $(v, \emptyset)\ \mathsf{if} \Rightarrow (\{(\mathsf{t}, \emptyset)\}, \{(v, \emptyset)\}, \emptyset)$,
6. (*If-*f) $(v, \emptyset)\ \mathsf{if} \Rightarrow (\{(\mathsf{f}, \emptyset)\}, \emptyset, \{(v, \emptyset)\})$,
7. (*If-*$\star$) $(\star, \emptyset)\ \mathsf{if} \Rightarrow (\{(\star, \emptyset)\}, \emptyset, \emptyset)$.

To illustrate the transition rules of the automaton, let t be an input Σ-binding tree.

• Assume the current state is (v, ρ) at node n of t, and n is labelled $\lambda\overline{\xi}$. If $n1$ is labelled with variable x_i and $m \curvearrowleft n1$ then m is labelled $\lambda x_1 \cdots x_k$ for some k and the state at m has the form (u, ρ_0). One of the profiles for x_i, say $(x_i, v, \rho_1) \in \rho_0$, where the value v is the same as the value at n, is chosen and the state at $n1$ is (v, ρ_1).

• If the state is (v, ρ) at node n of t and n is labelled $\xi^{(B_1,\cdots,B_m,o)}$, then ρ consists of profiles of ξ^{B_i} where $1 \leq i \leq m$, reflecting the eventual return of the traversal to the successors of n.

• If the state is $(\star, \emptyset)$ at node n of t, and n is labelled if, then each of rules (*If*-t), (*If*-f) and (*If*-$\star$) is applicable, reflecting the three cases in which a term $(\mathsf{if}\, B\, P\, Q)$ can evaluate to $\star$: B evaluates to t and P evaluates to $\star$; B evaluates to f and Q evaluates to $\star$; or B evaluates to $\star$.

Lemma 13: Let Σ be Σ_Ξ (respectively $\Sigma_\Xi \backslash \{\star\}$), $v \in \mathit{Val}$, and t a Σ-binding tree accepted by the traversal-simulating ADTA $\mathsf{A}(\Sigma, (v, \emptyset))$. Then t is well-typed. Hence, by Lemma 10, t represents a closed fPCF*- (respectively fPCF-) term.

Example 4.4: Consider the program $MN : o$ as defined in Example 3.4. Figure 3 gives an accepting run-tree of the ADTA $\mathsf{A}(\Sigma_\Xi, (\star, \emptyset))$ over the Σ_Ξ-binding tree $\lambda(MN)$ (which is precisely $\lambda_\Xi(MN)$).

Proposition 14: Let $\Xi : A$ be a singleton variable assignment where $A = (A_1, \cdots, A_n, o)$, $P := M\overline{N}$ a fPCF**-program where $M : A$ and $N_i : A_i$ are β-NFs, and $v \in \mathit{Val}$.*

(i) If there is an accepting run-tree of the ADTA $\mathsf{A}(\Sigma_\Xi, (v, \emptyset))$ over $\lambda_\Xi(P)$, then there is a v-complete traversal over the full binding tree $\lambda^{\mathrm{f}}_\Xi(P)$.

(ii) Suppose $\mathbf{t}$ is a v_0-complete traversal over $\lambda^{\mathrm{f}}_\Xi(P)$ and let $\mathcal{P}$ be the set of P-views $\ulcorner u \urcorner$ as u ranges over prefixes of $\widehat{\mathbf{t}}$. Each element of $\mathcal{P}$ is a finite sequence of the shape $t\, w$ where the question-prefix t consists of only question nodes, and the answer-suffix w, which is of length at most 2, consists of only answer nodes. Then the set of question-prefixes of elements of $\mathcal{P}$ is an accepting run-tree of the ADTA $\mathsf{A}(\Sigma_\Xi, (v_0, \emptyset))$ over the Σ_Ξ-binding tree $\lambda_\Xi(P)$.

Theorem 15: Let $\Xi : A$ be a singleton variable assignment with $A = (A_1, \cdots, A_n, o)$, $v \in \mathit{Val}$, and M a closed fPCF**-term in β-NF. The following are equivalent.*

(i) $M \in v$-REACH $[\text{fPCF}^\star, \text{fPCF}]$.

(ii) There exist closed fPCF*-terms $N_1 : A_1, \cdots, N_n : A_n$ (in β-NF) such that there is a v-complete traversal over the full computation tree $\lambda^{\mathrm{f}}_\Xi(M\overline{N})$,*

(iii) There exist closed fPCF*-terms $N_1 : A_1, \cdots, N_n : A_n$ (in β-NF) such that there is an accepting run-tree of the ADTA $\mathsf{A}(\Sigma_\Xi, (v, \emptyset))$ over the computation tree $\lambda_\Xi(M\overline{N})$.*

(iv) There exists $\rho = \bigcup_{i=1}^n \rho_i$ where $\rho_i = \{(\xi^{A_i}, v_{ij}, \rho_{ij}) : 1 \leq j \leq r_i\} \subseteq \mathbf{VP}_{\Xi_i}$ with $r_i \geq 0$ and $\Xi_i : A_i$ being the obvious restriction of Ξ to A_i, such that:
(1) $\mathsf{A}(\Sigma_\Xi, (v, \rho))$ accepts $\lambda_\Xi(M)$, and

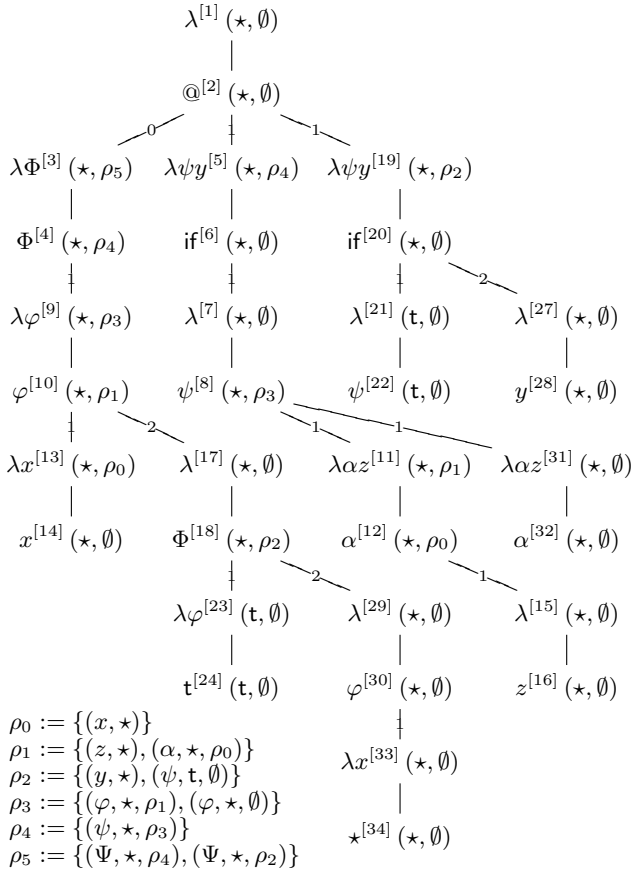

Figure 3. An accepting run-tree of $\mathsf{A}(\Sigma_\Xi, (\star, \emptyset))$ over $\lambda(MN)$ as defined in Example 3.4. (The superscripts attached to the node labels refer to the corresponding $\star$-complete traversal.)

(2) for each $1 \leq i \leq n$ the language of Σ_i-binding trees recognised by the intersection automaton $\prod_{j=1}^{r_i} \mathsf{A}(\Sigma_i, (v_{ij}, \rho_{ij}))$ — which is an ADTA by Lemma 12 — is non-empty, where $\Sigma_i = \Sigma_{\Xi_i} \backslash \{\star\}$.

Proof: That (i) and (ii) are equivalent is an immediate consequence of the definition of the decision problem, computational adequacy of innocent game semantics for fPCF* (by a proof similar to that of [8, Proposition 7.5, p. 373]), and Theorem 11. That (ii) and (iii) are equivalent follows from Proposition 14. Unpacking the definition of run-tree over the Σ_Ξ-binding tree $\lambda_\Xi(M\overline{N})$, statement (iii) is equivalent to: there exist closed fPCF-terms $N_1 : A_1, \cdots, N_n : A_n$, and $\rho = \bigcup_{i=1}^n \rho_i$ where $\rho_i = \bigcup_{j=1}^{r_i} \{(\xi^{A_i}, v_{ij}, \rho_{ij})\} \subseteq \mathbf{VP}_{\Xi_i}$ and $r_i \geq 0$ such that: (a) there is an accepting run-tree of $\mathsf{A}(\Sigma_\Xi, (v, \rho))$ over $\lambda_\Xi(M)$, and (b) for each $1 \leq i \leq n$ and $1 \leq j \leq r_i$, there is an accepting run-tree of $\mathsf{A}(\Sigma_i, (v_{ij}, \rho_{ij}))$ over $\lambda_{\Xi_i}(N_i)$. Thanks to Lemma 13 it is straightforward to see that the preceding statement is equivalent to (iv). ■

Stirling has asked [22, §5]: "There are significant open questions for the alternating [dependency tree] automata: are

they more expressive than the non-deterministic automata (NDTA), and is their non-emptiness problem decidable?". We can now answer both questions.

Corollary 16: The ADTA non-emptiness problem is undecidable. Since the non-emptiness problem is decidable for NDTA [22], ADTA are more expressive than NDTA.

Proof: Suppose the ADTA non-emptiness problem is decidable. It then follows from the equivalence of (iv) and (i) of Theorem 15 that the problem t-REACH $[\text{fPCF}^\star, \text{fPCF}]$ is decidable, which contradicts Corollary 7 (i). ■

A relativised version of reachability

An ***alternating*** Σ***-tree automaton*** (or ATA for short) is a quadruple $\langle\, Q, \Sigma, q_0, \Delta \,\rangle$ where Q is a finite set of states, Σ is a finite ranked alphabet, $q_0 \in Q$ is the initial state, and Δ is a finite set of transition rules of the form:

$$q\, s \;\Rightarrow\; (Q_1, \cdots, Q_k)$$

where $s \in \Sigma$ with $ar(s) = k$, $q \in Q$ and $Q_1, \cdots, Q_k \subseteq Q$.

A ***run-tree*** of an ATA $\langle\, Q, \Sigma, q_0, \Delta \,\rangle$ on a Σ-tree t is a $(\Sigma \times Q)$-labelled (unranked) tree whose nodes are pairs of the form (n, α) where n is a node of t and $\alpha \in Q^+$ is a non-empty sequence of states, labelled (s, q) if n is labelled s in t and q is the last element of the sequence α, which is defined top-down with root node (ε, q_0) where ε is the root node of t. Consider a node (n, α) of a partial run-tree which does not have successors. If $q\, s \Rightarrow (Q_1, \cdots, Q_k) \in \Delta$ then the successors of (n, α) are $(ni, \alpha q')$, for each $1 \leq i \leq k$ and $q' \in Q_i$. A run-tree of A on t is *accepting* if whenever (s, q) is a label of a leaf of the run-tree then either s has arity 0 or $q\, s \Rightarrow (\emptyset, \cdots, \emptyset)$ is a Δ-rule. The ATA A ***accepts*** a Σ-tree t if there is an accepting run-tree of A on t.

Definition 4.5: Let Ξ be a variable assignment of type $A = (A_1, \cdots, A_n, o)$ and Σ be Σ_Ξ or $\Sigma_\Xi \setminus \{\star\}$. The ***traversal-simulating ATA*** $\mathsf{B}(\Sigma, q_0) := \langle\, Q, \Sigma, q_0, \Delta \,\rangle$ where $Q = \mathit{Val} \times \mathcal{P}(\mathbf{VP}_\Xi) \times \mathcal{P}(\mathbf{VP}_\Xi)$, and the transition relation Δ is defined as follows (let $B = (B_1, \cdots, B_m, o) < A$).

1. $(v, \emptyset, \emptyset)\, @^{A \to A} \;\Rightarrow\; (\{(v, \rho, \emptyset)\}, Q_1, \cdots, Q_n)$ where $\rho \subseteq \bigcup_{i=1}^{n} \mathbf{VP}_\Xi(A_i)$ and $Q_i = \{(u, \rho', \emptyset) : (\xi^{A_i}, u, \rho') \in \rho\}$
2. $(v, \rho, \pi)\, \xi^B \Rightarrow (Q_1, \cdots, Q_m)$ where $\pi = \{(\xi^B, v, \rho)\} \cup \bigcup_{i,j=1,1}^{m,r_i} \pi_{ij}$, $\rho = \bigcup_{i,j=1,1}^{m,r_i} \{(\xi_i^{B_i}, v_{ij}, \rho_{ij})\}$ and $Q_i = \bigcup_{j=1}^{r_i} \{(v_{ij}, \rho_{ij}, \pi_{ij})\}$
3. $(v, \rho, \pi)\, \lambda\overline{\xi} \Rightarrow (v, \rho_1, \rho \cup \pi)\, \varphi$ where $(\varphi, v, \rho_1) \in \rho \cup \pi$
4. $(v, \rho, \pi)\, \lambda\overline{\xi} \Rightarrow (v, \emptyset, \rho \cup \pi)\, \$$ where $\$ \in \{\mathsf{if}, v\}$
5. $(v, \emptyset, \pi_1 \cup \pi_2)\, \mathsf{if} \Rightarrow (\{(\mathsf{t}, \emptyset, \pi_1)\}, \{(v, \emptyset, \pi_2)\}, \emptyset)$
6. $(v, \emptyset, \pi_1 \cup \pi_2)\, \mathsf{if} \Rightarrow (\{(\mathsf{f}, \emptyset, \pi_1)\}, \emptyset, \{(v, \emptyset, \pi_2)\})$
7. $(\star, \emptyset, \pi)\, \mathsf{if} \Rightarrow (\{(\star, \emptyset, \pi)\}, \emptyset, \emptyset)$

The ATA $\mathsf{B}(\Sigma, (v, \rho, \pi))$ simulates traversals in the same way as ADTA $\mathsf{A}(\Sigma, (v, \rho))$. Since ATA is defined over Σ-trees (as opposed to Σ-binding trees), it relies on the symbols in Σ_λ and Σ_{var} to work out the binding relation. Consequently an ATA state has a third component of *environment* π, consisting of profiles of variables that are currently within the scope of an enclosing binder.

Remark 4.6: The ATA in Definition 4.5 has much in common with the traversal-simulating alternating parity tree automaton of Ong [19] (even though the latter is an accepting devise of *infinite* ranked trees generated by higher-order recursion schemes). The state of the former, which has the form (v, ρ, π), is a more compact representation of the state of the latter, which has the form $v\, \pi\, (\varphi, v, \rho)$.

Let $\Xi : A$ be a variable assignment. A closed $\text{fPCF}^\star$-term $M : B \leq A$ in $\beta\eta$-long normal form ($\beta\eta$-LNF) is ***consistent*** with Ξ if $\lambda(M)$ is a Σ_Ξ-tree.[13] Let $\text{fPCF}^\star_\Xi$ be the set of such terms M, and fPCF_Ξ be its $\star$-free restriction. The following theorem is proved using the same argument as in the proof of Theorem 15, via the ATA analogue of Proposition 14.

Theorem 17: Let $\Xi : A$ be a variable assignment with $A = (A_1, \cdots, A_n, o)$, $\Xi_i = \Xi \restriction A_i$, $v \in \mathit{Val}$, and M a closed $\text{fPCF}^\star_\Xi$-term in $\beta\eta$-LNF. The following are equivalent.

(i) $M \in v$-REACH $[\text{fPCF}^\star_\Xi, \text{fPCF}_\Xi]$.

(ii) There exist closed fPCF_Ξ-terms $N_1 : A_1, \cdots, N_n : A_n$ (in $\beta\eta$-LNF) such that there is a v-complete traversal over the full computation tree $\lambda^{\mathrm{f}}(M\overline{N})$.

(iii) There exist closed fPCF_{Ξ_i}-terms $N_1 : A_1, \cdots, N_n : A_n$ (in $\beta\eta$-LNF) such that there is an accepting run-tree of the ATA $\mathsf{B}(\Sigma_\Xi, (v, \emptyset, \emptyset))$ over the computation tree $\lambda(M\overline{N})$.

(iv) There exists $\rho = \bigcup_{i=1}^{n} \rho_i$ where $\rho_i = \{(\xi^{A_i}, v_{ij}, \rho_{ij}) : 1 \leq j \leq r_i\} \subseteq \mathbf{VP}_{\Xi_i}$ and $r_i \geq 0$, such that:
   (1) $\mathsf{B}(\Sigma_\Xi, (v, \rho, \emptyset))$ accepts $\lambda(M)$, and
   (2) for each $1 \leq i \leq n$, setting $\Sigma_i = \Sigma_{\Xi_i} \setminus \{\star\}$, the language of Σ_i-trees recognised by the intersection automaton $\prod_{j=1}^{r_i} \mathsf{B}(\Sigma_i, (v_{ij}, \rho_{ij}, \emptyset))$ is non-empty.

Alternating Σ-tree automata are closed under intersection, and their acceptance problem is decidable. Further, their non-emptiness problem is also decidable; and if an alternating Σ-tree automaton has a non-empty language, a Σ-tree accepted by it can be effectively constructed (see for example the draft book [6]). Thus we can conclude:

Corollary 18 (Decidability): For every variable assignment $\Xi : A$, $v \in \mathit{Val}$, and closed $\text{fPCF}^\star_\Xi$-term $M : A$ in $\beta\eta$-LNF, the set $\{\overline{N} \in (\text{fPCF}_\Xi)^n : N_i$ in $\beta\eta$-LNF$, M\overline{N} \twoheadrightarrow v\}$ is regular. Hence, the relativised decision problem v-REACH $[\text{fPCF}^\star_\Xi, \text{fPCF}_\Xi]$ is decidable.

Corollary 19: The problems $\star$-REACH $[\text{fPCF}^\star, \text{fPCF}]$ and $\star$-REACH $[\text{fPCF}^\star_\perp, \text{fPCF}]$ are decidable at order 3.

Proof: We consider only $\star$-REACH $[\text{fPCF}^\star, \text{fPCF}]$. For $\star$-REACH $[\text{fPCF}^\star_\perp, \text{fPCF}]$, the same analysis as above can lead to an analogue of Theorem 17 where principal terms are taken from $\text{PCF}^\star_{\perp\,\Xi}$, and thus an argument similar to the following applies. It suffices to show that,

[13] Not all closed $\beta\eta$-LNFs $M : B \leq A$ are consistent with (singleton) $\Xi : A$. For example, each term M_n in Remark 3.2 is inconsistent with every assignment Ξ such that $\Xi^{(o,o)}$ contains less than n names.

given a closed fPCF*-term M in $\beta\eta$-LNF of order-3 type $A = (A_1, \cdots, A_n, o)$, we can find an assignment $\Xi : A$ such that $M \in \star$-REACH $[\text{fPCF}^\star, \text{fPCF}]$ iff $M \in \star$-REACH $[\text{fPCF}^\star_\Xi, \text{fPCF}_\Xi]$. For this, it suffices to find an assignment $\Xi : A$ such that every (up to α-equivalence) closed $\beta\eta$-LNF $N_i : A_i$ is consistent with $\Xi_i = \Xi \restriction A_i$. In fact, it suffices to show that every such N_i is consistent with some singleton assignment $\Xi_i : A_i$. Take $N_i : A_i = (B_1, \cdots, B_m, o)$ of order at most 2. Then $N_i = \lambda x_1^{B_1} \cdots x_m^{B_m}.N'$ for some $x_1, \ldots, x_m$ and open term $N' : o$ whose free variables are contained in the x_j's. We show that N' cannot contain λ-abstractions and therefore all its variables (be they free or bound) are within the x_j's; thus N_i is consistent with any singleton $\Xi_i : A_i$. Indeed, since N_i is closed and in $\beta\eta$-LNF, N' must be of the form $\alpha\,\overline{L}$, where α is either a constant or some x_j (and thus of order at most 1). If $\alpha : o$ then we are done; otherwise, we repeat the same argument for each $L_k : o$. ■

Connections to observational equivalence

Consider the following generalisation of our reachability template, where $\bar\theta$ is a sequence $\theta_1, \cdots, \theta_m$ of $\mathcal{L}_1$-values.

> $\bar\theta$-REACH $[\mathcal{L}_1, \mathcal{L}_2]$: Given closed $\mathcal{L}_1$-terms $M_1, \cdots, M_m$ of type $(A_1, \cdots, A_n, o)$, are there closed $\mathcal{L}_2$-terms $N_1, \cdots, N_n$ s.t. $M_i\overline{N} \twoheadrightarrow \theta_i$ for each $1 \leq i \leq m$?

The generalised decision problem can be characterised by use of ADTA's and ATA's (in its relativised version) in exactly the same fashion as its one-dimensional counterpart (via analogues of Theorems 15 and 17). On the other hand, given fPCF$_\perp$-terms $M, M' : A$, $M \not\lesssim M'$ iff (M, M') belongs to (θ_1, θ_2)-REACH $[\text{fPCF}_\perp, \text{fPCF}_\perp]$ where (θ_1, θ_2) is one of $(\mathsf{t}, \perp), (\mathsf{t}, \mathsf{f}), (\mathsf{f}, \perp)$ and (f, t). This gives us a means to relate these classes of automata with observational approximation and equivalence of fPCF$_\perp$ (and of other extensions of fPCF). For example, observational approximation of third-order fPCF$_\perp$ can be decided by use of ATA's.

V. FURTHER DIRECTIONS

(i) Variable profiles are a useful construct for an automata-theoretic simulation of traversals, which are an accurate model of control flow. It would be interesting to investigate possible applications of variable profiles to CFA. (ii) The MSO theory of higher-order recursion schemes (HORS) is decidable. An alternative approach to functional reachability is to cast the problem in the setting of HORS extended with booleans. As the extension can be CPS-transformed out, one can then use Ong's model-checking algorithm [19] or Kobayashi and Ong's method based on intersection types [10], [12], [11]. (iii) Finally, an obvious open problem is the following.

Conjecture 20: $\star$-REACH $[\text{fPCF}^\star, \text{fPCF}]$ *is decidable.*

REFERENCES

[1] T. Ball and S. K. Rajamani. The SLAM Project: Debugging system software via static analysis. In *Proc. POPL*, 2002.

[2] S. Berardi, M. Coppo, F. Damiani, and P. Giannini. Type-based useless-code elimination for functional programs. In *Proc. SAIG*, 2000.

[3] W. Blum. *The Safe Lambda Calculus*. PhD thesis, University of Oxford, Dec 2008.

[4] W. Blum and C.-H. L. Ong. Local computation of β-reduction: a concrete presentation of game semantics. 2009. Preprint, downloadable at william.famille-blum.org/research/.

[5] R. Cartwright, P.-L. Curien, and M. Felleisen. Fully abstract semantics for observably sequential languages. *Information and Computation*, 111:297–401, 1994.

[6] H. Common, M. Dauchet, R. Gilleron, F. Jacquemard *et al. Tree Automata Techniques and Applications*. 2007. Draft book.

[7] T. A. Henzinger, R. Jhala, R. Majumdar, and G. Sutre. Temporal-safety proofs for systems code. In *Proc. SPIN Workshop*, 2003.

[8] J. M. E. Hyland and C.-H. L. Ong. On Full Abstraction for PCF: I,II,III.. *Info. and Comp.*, 163:285–408, 2000.

[9] N. Kobayashi. Type-based useless-variable elimination. *Higher Order Symbolic Computation*, 14(2-3):221–260, 2001.

[10] N. Kobayashi. Types and higher-order recursion schemes for verification of higher-order programs. In *Proc. POPL*, 2009.

[11] N. Kobayashi and C.-H. L. Ong. Complexity of model checking recursion schemes for fragments of the modal mu-calculus. In *Proc. ICALP*, 2009.

[12] N. Kobayashi and C.-H. L. Ong. A type theory equivalent to the modal mu-calculus model checking of higher-order recursion schemes. In *Proc. LICS*, 2009.

[13] R. Loader. Finitary PCF is not decidable. *Theoretical Computer Science*, 266:342–364, 2001.

[14] P. Malacaria and C. Hankin. Generalised flowcharts and games. In *Proc. ICALP*, 1998.

[15] P. Malacaria and C. Hankin. A new approach to control flow analysis. In *Proc. Compiler Construction*, 1998.

[16] J. Midtgaard. Control-flow analysis of functional programs. Tech. Report BRICS RS-07-18, DAIMI, U. of Aarhus, 2007.

[17] R. Milner. Fully abstract models of typed lambda-calculus. *Theoretical Computer Science*, 4:1–22, 1977.

[18] F. Nielson, H. R. Nielson, and C. Hankin. *Principles of Program Analysis*. Springer-Verlag New York, 1999.

[19] C.-H. L. Ong. On model-checking trees generated by higher-order recursion schemes. In *Proc. LICS*, 2006. Long version downloadable at users.comlab.ox.ac.uk/luke.ong/.

[20] G. D. Plotkin. LCF considered as a programming language. *Theoretical Computer Science*, 5:223–255, 1977.

[21] K. Sieber. Reasoning about sequential functions via logical relations. In *Proc. Applic. of Categories in Comp. Sc.*, 1992.

[22] C. Stirling. Dependency tree automata. In *Proc. FoSSaCS*, 2009.

[23] M. Zaionc. On the lambda definable higher-order boolean functionals. *Fundamenta Informaticae*, 12, 1989.