Theoretical Computer Science 266 (2001) 341–364

Theoretical Computer Science

www.elsevier.com/locate/tcs

Finitary PCF is not decidable

Ralph Loader

Merton College, Oxford, UK

Received October 1996; revised March 2000; accepted April 2000

Communicated by G.D. Plotkin

Abstract

The question of the decidability of the observational ordering of finitary PCF was raised (Jung and Stoughton, in: M. Bezem, J.F. Groote (Eds.), Typed Lambda Calculi and Applications, Lecture Notes in Computer Science, vol. 664, Springer, Berlin, 1993, pp. 230–244) to give mathematical content to the full abstraction problem for PCF (Milner, Theoret. Comput. Sci. 4 (1977) 1–22). We show that the ordering is in fact undecidable. This result places limits on how explicit a representation of the fully abstract model can be. It also gives a slight strengthening of the author's earlier result on typed λ-definability (Loader, in: A. Anderson, M. Zeleny (Eds.), Church Memorial Volume, Kluwer Academic Press, Dordrecht, to appear).

Keywords: PCF; Decidability; Lambda calculus; Full abstraction

0. Introduction

The language PCF was introduced by Plotkin [14] as a functional programming language simple enough to be analysed mathematically, developing on work of people such as Kleene and Scott. PCF is a typed λ-calculus with natural numbers and recursion at arbitrary types, and a call-by-name operational semantics. There is a natural ordering on the closed terms of this calculus, namely the *observational pre-order*, defined by setting $s \leqslant t$ whenever s and t are such that replacing s by t in any terminating program yields another terminating program.

A model is called fully abstract if it characterises observational equivalence, with any two terms s and t having equal interpretation in the model iff $s \leqslant t \leqslant s$. Milner [9] showed that there is (up to isomorphism) a unique interpretation of the types of PCF that is fully abstract and that satisfies some natural technical conditions (see [9] for details).

E-mail address: suckfish@ihug.co.nz (R. Loader).

0304-3975/01/$ - see front matter
PII: S0304-3975(00)00194-8

While that result uniquely characterises desirable models of PCF, it uses a term model construction that does not tell us what mathematical structures are appropriate for modelling the calculus, and does not give useful techniques for reasoning about the model. For example, from Milner's work we know that types are represented by some sort of domain, but it is hard to find an operation on domains mapping the interpretations of types A and B to the interpretation of the function type $A \Rightarrow B$ (it is easily shown not be any of the function spaces usually considered in domain theory).

The problem of finding a collection of mathematical structures giving rise to the fully abstract model is known as the full abstraction problem. The aim is to find a presentation of the fully abstract model that is, as far as possible, presented in a manner that is both concrete and independent of syntax. Recently, several solutions to this problem have been given [1, 4, 10, 11]. However, the problem as posed is not precisely defined; there is no clear mathematical definition of what separates Milner's syntactical construction from the more semantical constructions that the full abstraction problem asks for. In particular, it is not entirely clear in what sense, if any, the solutions mentioned above can be considered a best possible solution.

One property that could be required of a presentation of the fully abstract model, is to be given very concretely. Specifically, for 'finitary' parts of the model, one could require that the presentation involves only computable operations on finitely represented objects belonging to some decidable set. Some work in this direction was carried out in the 1980s and early 1990s, producing refined versions of continuous functions on domains, such as stable and strongly stable functions.

While the observational ordering of PCF is clearly undecidable, giving a concrete presentation as above would show decidability for the restriction of the observational ordering to certain 'finitary' terms. The issues above led Jung and Stoughton [5] to ask if this restriction of the observational ordering is in fact decidable, both as a test of individual solutions to the full abstraction problem, and as a test for the solvability of the problem.

We show that this is not the case. The observational ordering of the finitary parts of PCF is undecidable.

The finitary terms of PCF can be given by calculi known as finitary PCF, where instead of having a type for all natural numbers, we have a type of n distinct values, for some natural number n. Here, we consider PCF_2, which has two just values, written as `tt` and `ff`. Our proof of undecidability proceeds via an encoding of semi-Thue problems. The main difficulty is that because of the rich term structure of finitary PCF, a series of reductions of complicated terms to simpler ones must be carried out, and the encoding used must be carefully chosen so as to make this possible.

0.1. Notations and conventions

We mention some notational conventions used in this paper. Variables of a λ-calculus are denoted using a single typewriter style letter, `x`, `i`, `W`, ..., possibly with a super- or sub-script. When we write a term, we assume that different letters always represent different variables. We take the λ-abstraction notation $\lambda\mathtt{x}\,.\,s$ to be a meta-notation for an α-equivalence class (or terms with de Bruijn indices), so that $\lambda\mathtt{x}\,.\,\mathtt{x} = \lambda\mathtt{y}\,.\,\mathtt{y}$. Variable are typed, so that $\lambda\mathtt{x}\,.\,\mathtt{x}$ has a unique type (which depends on the variable $\mathtt{x}$), but the types are not mentioned when they are clear from the context. In particular, letters near the end of the alphabet, $\mathtt{x}$ ... are usually taken to have the ground type $\mathscr{B}$. Whenever we write a term, we assume it to be well-typed.

An overline ($\bar{x}$) is used to denote a finite sequence $(x_1, x_2 \ldots)$. If f is some function or operation, then $f\,\bar{x}$ is used to represent the sequence $f x_1, f x_2 \ldots$, with the exception that if f and the $\bar{x}$ are terms then $f\,\bar{x}$ represents repeated application, $f\,x_1\,x_2 \ldots$. If we write a sequence $\bar{x}$ as $x_1 \ldots x_e$, where e is some expression whose value is not otherwise defined, we take this to be implicitly defining e to be the length of $\bar{x}$.

A term t may be written with some variables displayed: $t[\mathtt{x}_1 \ldots \mathtt{x}_n]$. When this is done, $t[a_1 \ldots a_n]$ means the result of substituting the $\bar{a}$ for the $\bar{\mathtt{x}}$ in t. A *substitution* is a function σ mapping variables to terms and preserving types. If t is a term, then $\sigma\,t$ denotes the result of substituting $\sigma\,\mathtt{x}$ for each free variable occurrence $\mathtt{x}$ in t.

We use semi-Thue systems over the *alphabet* $\{\mathtt{tt}, \mathtt{ff}\}$. *Words* are finite, non-empty sequences of `tt`s and `ff`s. *Rules* are pairs of words, written $[C \to C']$. Concatenation of sequences is denoted by juxtaposition. A *semi-Thue system* consists of an initial word W_0 and a finite set $\{R_1 \ldots R_N\}$ of rules. Note that a word W always has non-zero length, however, if we write W in the form $D_1\,C\,D_2$, there is no implicit requirement that D_1, C and D_2 all have non-zero length.

A derivation step by a rule $R = [C \to C']$ consists of a pair of words in the form $(D_1 C D_2, D_1 C' D_2)$. A derivation in a semi-Thue system consists of a finite sequence $W_0, W_1, \ldots, W_p$, where W_0 is the initial word, and each pair (W_{i-1}, W_i) is a derivation step by some rule of the system. A word W is said to be *derivable* if there is a derivation ending with W.

It is well known (see [8] for a recent proof) that there is a semi-Thue system for which the derivability predicate is undecidable. We fix such a system $W_0, R_1, \ldots, R_N$ throughout this paper.

We shall deal with encodings mapping semi-Thue systems into the syntax of finitary PCF. Such encodings are denoted by typewriter style words with an initial capital, such as `Enc` and `PosCh`.

1. Preliminaries

Definition 1. *Finitary PCF* (PCF_2) is the simply typed λ-calculus, with a single ground type $\mathscr{B}$, three constants of ground type: `tt`, `ff` and $\perp$, and a ternary construct, `if` ... `then` ... `else` ... taking three terms of type $\mathscr{B}$ and producing another of the same type.

The intention is that `tt` and `ff` represent two distinct, discrete values, which may as well be taken to be the usual two Boolean values, while $\perp$ represents non-termination.

We use the β-reduction and η-expansion of the simply typed λ-calculus, and add the following reductions for the new constructs:

$$\begin{aligned} \texttt{if tt then } a \texttt{ else } b &\rightarrow a, \\ \texttt{if ff then } a \texttt{ else } b &\rightarrow b, \\ \texttt{if } \bot \texttt{ then } a \texttt{ else } b &\rightarrow \bot, \end{aligned}$$

and

$$\begin{gathered} \texttt{if (if } a \texttt{ then } b \texttt{ else } c\texttt{) then } d \texttt{ else } e, \\ \downarrow \\ \texttt{if } a \texttt{ then (if } b \texttt{ then } d \texttt{ else } e \texttt{ else (if } c \texttt{ then } d \texttt{ else } e). \end{gathered}$$

As the calculus is strongly normalising and Church-Rosser (this can be shown by any number of standard techniques; general results covering our calculus can be found in [3]), there is no need here to be overly concerned with a precise presentation of the operational semantics.

We note briefly the shape of the normal forms of the reductions above:

- The normal forms of a type $A_1 \Rightarrow \cdots \Rightarrow A_n \Rightarrow \mathscr{B}$ are $\mathtt{x}_1 \ldots \mathtt{x}_n . r$ where $r : \mathscr{B}$ is normal.
- tt, ff and $\bot$ are normal.
- If f is a variable of type $A_1 \Rightarrow \cdots \Rightarrow A_n \Rightarrow \mathscr{B}$ $(n \geqslant 0)$ and the s_i are normal of type A_i, and $b, c : \mathscr{B}$ are normal, then $\mathtt{f}\, s_1 \ldots s_n$ and $\texttt{if f}\,\bar{s}\texttt{ then } b \texttt{ else } c$ are normal.

In particular, the closed normal forms of type $\mathscr{B}$ are just tt, ff and $\bot$.

Definition 2. We define the *observational pre-order* $\leqslant$ as follows:

- $x \leqslant y$ iff either $x = \bot$ or $y = x$, for $x, y \in \{\mathtt{tt}, \mathtt{ff}, \bot\}$.
- $\leqslant$ is extended to closed terms of type $\mathscr{B}$ by comparing normal forms.
- $\leqslant$ is extended to closed terms of a function type $A \Rightarrow B$ by $f \leqslant g$ iff

$$f x \leqslant g y \quad \text{whenever} \quad x \leqslant y.$$

(In other words, $\leqslant$ is a *logical relation*.)

Observational equivalence is defined by $a \equiv b$ iff $a \leqslant b$ and $b \leqslant a$. It is a logical equivalence relation (see below).

The *minimal extensional* or *fully abstract* model of PCF_2 is defined by interpreting each type A as the quotient by $\equiv$ of the closed terms of type A.

The relations $\leqslant$ and $\equiv$ are extended to non-closed terms, by putting $s \leqslant t$ if and only if $\sigma s \leqslant \sigma t$ whenever σ is a substitution of closed terms for the free variables of s and t; this is equivalent to comparing appropriate closures of s and t.

The aim of this article is to show that the relations $\leqslant$ and $\equiv$ are undecidable.

As $\leqslant$ is a logical relation and a pre-order at type $\mathscr{B}$, certain properties of $\leqslant$ and $\equiv$ follow from the logical relations lemma [13, 17]:

Lemma 3. *$\leqslant$ is a pre-order, and $\equiv$ is an equivalence, at all types. If $f, g : A \Rightarrow B$ then $f \leqslant g$ iff $f x \leqslant g x$ for all closed $x : A$.*

The relations $\leqslant$ and $\equiv$ contain the conversion relation $\to$.

If $C[\cdot]$ is a context with a hole of type A, and $s \leqslant t$ are terms of type A, then $C[s] \leqslant C[t]$.

These properties suffice to show that the definition above of the observational pre-order gives the same relation as the usual definition that uses contexts and operational semantics.

Lemma 4. *At each type there are only finitely many $\equiv$ classes of closed terms.*

Proof. By induction on types. The only closed equivalence classes at type $\mathscr{B}$ are those of $\mathtt{tt}$, $\mathtt{ff}$ and $\perp$. The equivalence class of a closed term f of type $A \Rightarrow B$ is determined by the function $[x] \mapsto [fx]$ it induces from the equivalence classes of type A to those of type B. Therefore, if A and B have $|A|$ and $|B|$ many equivalence classes, then $A \Rightarrow B$ has no more than $|B|^{|A|}$ equivalence classes. $\square$

Lemma 5. *If $\equiv$ is a decidable relation, then so is the solvability (for X by a closed term) of systems of equations in the form*

$$\begin{aligned} X a_1^1 \dots a_n^1 &\equiv b^1 \\ &\vdots \\ X a_1^m \dots a_n^m &\equiv b^m. \end{aligned}$$

where each a_j^i is closed of type A_j and each b^i is either $\mathtt{tt}$ or $\mathtt{ff}$.

Proof. There is a term $G : \mathscr{B} \Rightarrow \cdots \Rightarrow \mathscr{B} \Rightarrow \mathscr{B}$ such that, for closed $x^1 \dots x^m$, if $x^i \equiv b^i$ $(1 \leqslant i \leqslant m)$, then $G x^1 \dots x^m \equiv \mathtt{tt}$, and else $G x^1 \dots x^m \equiv \perp$. Define $F : (A_1 \Rightarrow \cdots \Rightarrow A_n \Rightarrow \mathscr{B}) \Rightarrow \mathscr{B}$ to be

$$\lambda \mathtt{X} . G (\mathtt{X} a_1^1 \dots a_n^1) \dots (\mathtt{X} a_1^m \dots a_n^m).$$

Clearly $FX \equiv \mathtt{tt}$ when X satisfies the given equations, and $FX \equiv \perp$ otherwise. Hence $F \equiv \lambda \mathtt{X} . \perp$ if and only if the given equations have no solution. $\square$

Corollary 6. *If $\equiv$ is decidable, then so is the solvability (for X by a closed term) of systems of inequations in the form*

$$\begin{aligned} X a_1^1 \dots a_n^1 &\geqslant \beta^1 \\ &\vdots \\ X a_1^m \dots a_n^m &\geqslant \beta^m, \end{aligned}$$

where $a_j^i : A_j$ and $\beta^i : \mathscr{B}$ are closed.

Proof. Each β^i is equivalent to some $b^i \in \{\mathtt{tt}, \mathtt{ff}, \bot\}$. If $b^i = \bot$, then the ith inequation is always satisfied and may be discarded. If $b^i \in \{\mathtt{tt}, \mathtt{ff}\}$, then $x \geqslant \beta^i$ iff $x \equiv b^i$, and the ith inequality may be replaced by an equality of the form used in Lemma 5. This reduces the corollary to the lemma. $\square$

We shall encode semi-Thue systems in such a way that the above lemma may be applied. We shall use several (32) encodings simultaneously. Each encoding is a function mapping words to closed terms, satisfying the following conditions. The encodings are given in the appendix.

Definition 7 (*Word encoding*). A tt-*encoding* is a function Enc such that if W is a word, say with length n, then $\mathtt{Enc}(W)$ is a closed term of type $\underbrace{\mathscr{B} \Rightarrow \cdots \Rightarrow \mathscr{B}}_{2n+2} \Rightarrow \mathscr{B}$ such that,

$$\mathtt{Enc}(W) \leqslant \lambda \mathtt{x}_1 \ldots \mathtt{x}_{2n+2}.\mathtt{tt}.$$

The notion of ff-*encoding* is defined by replacing tt with ff above. A function Enc is called an *encoding* if either it is a tt-encoding, or it is a ff-encoding.

We also need encodings to map rules to closed terms. Instead of defining these explicitly, they are given using the following lemma.

Lemma 8. 1. *Suppose that $W = D_1 C D_2$, $W' = D_1 C' D_2$ are words where the lengths of D_1, D_2, C and C' are k_1, k_2, l and l', respectively. If* Enc *is an encoding, and the inequation* (1)

$$\begin{aligned} &\mathtt{Enc}\, W' \mathtt{x}_1 \ldots \mathtt{x}_{2k_1} \mathtt{y}'_1 \ldots \mathtt{y}'_{2l'} \mathtt{z}_1 \ldots \mathtt{z}_{2k_2} \mathtt{i}'\mathtt{j}' \\ &\quad \leqslant F\,(\lambda \mathtt{y}_1 \ldots \mathtt{y}_{2l} \mathtt{i}\mathtt{j}\,.\,\mathtt{Enc} W \overline{\mathtt{x}}\,\overline{\mathtt{y}}\,\overline{\mathtt{z}}\mathtt{i}\mathtt{j})\overline{\mathtt{y}}'\mathtt{i}'\mathtt{j}' \end{aligned} \tag{1}$$

holds both with $F = F_1$ and with $F = F_2$, then there is F with both $F \leqslant F_1$ and $F \leqslant F_2$ such that (1) *holds. This F depends only on F_1 and F_2.*

2. *Fix C and C'. Then there is a $\leqslant$-minimum closed F such that, for all D_1 and D_2, the inequation* (1) *holds.*

Proof. 1. A suitable F is given by

$$\lambda \mathtt{f}\overline{\mathtt{y}}'\mathtt{i}'\mathtt{j}'\,.\,g(F_1 \mathtt{f}\overline{\mathtt{y}}'\mathtt{i}'\mathtt{j}')(F_2\, \mathtt{f}\, \overline{\mathtt{y}}'\mathtt{i}'\mathtt{j}')$$

where $g\,a\,b$ is

$$\mathtt{if}\ a\ \mathtt{then}\ (\mathtt{if}\ b\ \mathtt{then\ tt\ else}\ \bot)\ \mathtt{else}\ (\mathtt{if}\ b\ \mathtt{then}\ \bot\ \mathtt{else\ ff}).$$

2. When Enc is a v-encoding, (1) is satisfied by $F = \lambda \mathtt{f}\, \overline{\mathtt{y}}'\, \mathtt{i}'\, \mathtt{j}'\,.\, v$. Because there are only finitely many $\equiv$ classes of closed terms at each type, this part is now immediate from the first. $\square$

Definition 9 (*Rule encoding*). Given an encoding $\mathtt{Enc}$, we define $\mathtt{Enc}\,R$ for a rule $R=[C \to C']$ to be the minimum F given by the second part of the lemma above. An encoding is called *exact* if this always gives equivalence in (1). Although all the encodings we shall consider are in fact exact, this plays no rôle in our arguments.

Note that the definition given of $\mathtt{Enc}[C \to C']$ is not effective; however, such effectiveness is not needed for the purposes in hand, since we only need consider finitely many rules and encodings. It is possible, although tedious, to calculate the required encoding of rules.

Lemma 10 (Soundness). *Suppose a word W is derivable from W_0 using the rules R_i. Then there is a term $t[\mathtt{W}_0, \mathtt{R}_1 \ldots \mathtt{R}_N, \mathtt{x}_1 \ldots \mathtt{x}_{2n+2}]$ with the indicated context, such that, for any encoding* $\mathtt{Enc}$,

$$\mathtt{Enc}\,W\,\overline{\mathtt{x}} \leqslant t[\mathtt{Enc}\,W_0, \mathtt{Enc}\,R_1 \ldots \mathtt{Enc}\,R_N, \overline{\mathtt{x}}]. \tag{2}$$

(*With equivalence holding if* $\mathtt{Enc}$ *is exact.*)

For a one step derivation, the lemma just restates the definition of the encoding of rules; an induction gives the general case.

Proof. By induction over the derivation of W. If $W=W_0$, then let

$$t[\mathtt{W}_0, \overline{\mathtt{R}}, \overline{\mathtt{x}}] = \mathtt{W}_0\overline{\mathtt{x}}.$$

For the induction step, suppose that $W=D_1CD_2$ is derivable, and that (2) holds. If $W'=D_1C'D_2$, where $R_j=[C \to C']$, then define

$$t'[\mathtt{W}_0, \overline{\mathtt{R}}, \overline{\mathtt{x}}, \overline{\mathtt{y}}', \overline{\mathtt{z}}, \mathtt{i}', \mathtt{j}'] = \mathtt{R}_j(\lambda\overline{\mathtt{y}}\mathtt{i}\mathtt{j}\,.\,t[\mathtt{W}_0, \overline{\mathtt{R}}, \overline{\mathtt{x}}, \overline{\mathtt{y}}, \overline{\mathtt{z}}, \mathtt{i}, \mathtt{j}])\overline{\mathtt{y}}'\mathtt{i}'\mathtt{j}',$$

where the notation is as in Lemma 8. The inequation (2), with W' and t' replacing W and t, now follows from the definition of $\mathtt{Enc}\,R_j$ (with equivalence holding if $\mathtt{Enc}$ is exact). $\square$

We now fix a finite set of encodings, as given in the appendix (pp. 00 and 00). We say that a term t *satisfies* a word W (w.r.t. $\mathtt{x}_1, \ldots, \mathtt{x}_{2n+2}$) if it is a normal term in context $\mathtt{W}_0$, $\overline{\mathtt{R}}$, $\overline{\mathtt{x}}$, and the inequation (2) holds for each of our encodings.

Lemma 11. *For any word W, there is a set of inequations (computable from W) in the form in Corollary 6, that is solvable if and only if W is satisfied by some term.*

Proof. Take all inequations in the form

$$X(\mathtt{Enc}\,W_0)(\mathtt{Enc}\,\overline{R})e_1 \ldots e_{2n+2} \geqslant \mathtt{Enc}\,We_1 \ldots e_{2n+2},$$

where $\mathtt{Enc}$ ranges over the 32 encodings in the appendix, and the e_i range over $\mathtt{tt}, \mathtt{ff}, \perp$.

Clearly, if t satisfies W, then $\lambda \mathtt{W}_0\,\overline{\mathtt{R}}\,\overline{\mathtt{x}}\,.\,t$ is a solution to the above inequations, while if F is a solution to the equations above, then $F\,\mathtt{W}_0\,\overline{\mathtt{R}}\,\overline{\mathtt{x}}$ satisfies W. $\square$

In the rest of the article, we establish a converse to the soundness lemma above: if a term t satisfies a word W, then the word W is derivable from W_0 using the rules R_i. By Corollary 6 and Lemma 11, this will suffice to establish our undecidability result. The proof of this is done in several stages. We take a term satisfying a word and simplify the term in several ways, arriving at a new term satisfying the same word, but also subject to severe structural constraints. An induction over the term then gives the result.

2. Descent

If a term t either in the form $\mathtt{R}_i f\,\overline{a}$ or $\mathtt{W}_0\,\overline{a}$ satisfies a word w.r.t. $\overline{\mathtt{x}}$, then we can deduce some properties of the $\overline{a}$ fairly straightforwardly. For example, by looking at the encoding $\mathtt{Fixtt}$, none of the a_i could be the constant $\mathtt{ff}$. However, we wish to make such deductions, not just at the ‘top level’ of the term, but at positions buried inside it. The material of this section gives us a framework for making such deductions.

The notions here are not especially difficult, but they are quite technical, and the reasons for introducing them may not be immediately obvious. The reader may find that a detailed reading of this section is easier if deferred until the content is used, later in our proof.

Definition 12. Let $\mathtt{Enc}$ be an encoding, and $R=[C\to C']$ be a rule, where C and C' have lengths l and l', respectively. If a sequence $g_1\ldots g_{2l+2}$ of closed terms of type $\underbrace{\mathscr{B}\Rightarrow\cdots\Rightarrow\mathscr{B}}_{2l'+2}\Rightarrow\mathscr{B}$ satisfies

$$\mathtt{Enc}\,R\leqslant\lambda\mathtt{f}\ \mathtt{x}_1\ldots\mathtt{x}_{2l'+2}\,.\,\mathtt{f}(g_1\overline{\mathtt{x}})\ldots(g_{2l+2}\overline{\mathtt{x}})$$

then we call $(g_1\ldots g_{2l+2})$ *descent functions* for $\mathtt{Enc}\,R$.

Expanding the definition of $\mathtt{Enc}\,R$, the statement that $g_1\ldots g_{2l+2}$ are descent functions for a $\mathtt{Enc}\,R$, is equivalent to the following: for any words $W=D_1CD_2$, $W'=D_1C'D_2$ and $\overline{x}\ \overline{y}'\overline{z}\,\alpha\,\beta$ such that

$$\mathtt{Enc}\,W'\overline{x}\,\overline{y}'\overline{z}\,\alpha\beta\equiv v$$

(where $\mathtt{Enc}$ is a v-encoding), we have also

$$\mathtt{Enc}\,W\overline{x}(g_1\overline{y}'\alpha\beta)\ldots(g_{2l}\overline{y}'\alpha\beta)\,\overline{z}\,(g_{2l+1}\overline{y}'\alpha\beta)(g_{2l+2}\overline{y}'\alpha\beta)\equiv v.$$

Lemma 13 (Descent existence). *For each of our encodings and each rule, there exist descent functions.*

Proof. In most cases, we can simply take appropriate constant functions. For encodings other than Lin and PosCh, if $R=[C\to C']$, and C is the word $c_1\dots c_l$, then descent functions are given by $g_{2i-1}\overline{x}=g_{2i}\overline{x}\equiv c_i$ $(1\leqslant i\leqslant l)$ and $g_{2l+1}\overline{x}=g_{2l+2}\overline{x}\equiv \mathtt{tt}$.

For the encoding Lin, it suffices to take the g to satisfy:

$$g_i x_1\dots x_{2l'+2}\equiv \mathtt{ff}\qquad (1\leqslant i\leqslant 2l+1),$$

$$g_{2l+2}\ \mathtt{ff}\dots\mathtt{ff}\equiv \mathtt{ff},$$

$$g_{2l+2}\underbrace{\mathtt{ff}\dots\mathtt{ff}}_{j-1}\ \mathtt{tt}\ x_{j+1}\dots x_{2l'+2}\equiv \mathtt{tt}\qquad (1\leqslant j\leqslant 2l'+2).$$

The remaining case of PosCh is omitted. □

Corollary 14 (Constant descent). *Suppose that the variables* $\overline{\mathtt{y}}$ *do not occur in* s. *Then, for each of our encodings, we have*

$$\mathtt{Enc}\,R(\overline{\mathtt{y}}\,.\,s)\overline{a}\leqslant s.$$

Proof. Take any descent functions $\overline{g}$. Then we have

$$\mathtt{Enc}\,R(\overline{\mathtt{y}}\,.\,s)\overline{a}\leqslant(\overline{\mathtt{y}}\,.\,s)(g_1\overline{a})\dots(g_{2l+2}\overline{a})\equiv s.\qquad \square$$

The use of descent functions will enable us to examine the behaviour of sub-terms in specific positions within a term. The collection of these sub-terms is defined below.

Definition 15. The *spinal sub-terms* of t are defined by:
- Any term is a spinal sub-term of itself.
- Any spinal sub-term of s is also a spinal sub-term of $\mathtt{R}_i(\overline{\mathtt{y}}.s)\overline{a}$.

Note that the variables free in a spinal sub-term r of t are either of type $\mathscr{B}$, or free in t.

The following lemma is immediate using repeated application of the definition of descent functions, once one unravels the notation. Most of its applications use constant descent functions.

Lemma 16 (Repeated descent). *Let* $t[\mathtt{W}_0,\overline{\mathtt{R}},\overline{\mathtt{x}}]$ *be a term. Let* Enc *be a* v*-encoding, and let* σ *be a substitution. Suppose that*

$$t[\mathtt{Enc}\,W_0,\mathtt{Enc}\,\overline{R},\sigma\overline{\mathtt{x}}]\equiv v$$

and that for every spinal sub-term in the form $\mathtt{R}_i\,(\lambda\overline{\mathtt{y}}\,.\,s)\,\overline{a}$ *there are* $\mathtt{Enc}\,R_i$ *descent functions* $\overline{g}$ *with* $\sigma\,\mathtt{y}_j\equiv g_j(\sigma\,\overline{a})$ *for each* j.

Then, for each spinal sub-term $r[\mathtt{W}_0,\overline{\mathtt{R}},\overline{\mathtt{x}},\overline{\mathtt{y}}]$, *we have*

$$r[\mathtt{Enc}\,W_0,\mathtt{Enc}\,\overline{R},\sigma\overline{\mathtt{x}},\sigma\overline{\mathtt{y}}]\equiv v.$$

Proof. By induction on t. If r is t, there is nothing to do. Otherwise, t is in the form $\mathtt{R}_i\,(\lambda\,\overline{y}\,.\,s)\,\overline{e}$, and r is a spinal sub-term of s. We have

$$\begin{aligned} v &\equiv t[\mathtt{Enc}\,W_0, \mathtt{Enc}\,\overline{R}, \sigma\overline{\mathtt{x}}] \\ &\leqslant s[\mathtt{Enc}\,W_0, \mathtt{Enc}\,\overline{R}, \sigma\overline{\mathtt{x}}, \overline{g}(\sigma\overline{e})] \\ &\equiv s[\mathtt{Enc}\,W_0, \mathtt{Enc}\,\overline{R}, \sigma\overline{\mathtt{x}}, \sigma\overline{\mathtt{y}}] \end{aligned}$$

using the fact that $\overline{g}$ are descent functions for the first step, and the relation between $\overline{g}$ and σ for the second step. Apply the induction hypothesis to s to obtain the result. $\square$

We state below an approximation to the encodings of rules (excepting `Lin`). It is useful in conjunction with repeated descent: if the sub-term r in the repeated descent lemma is in the form $\mathtt{R}_j\,f\,\overline{e}$, where R_j is $[C \to C']$, then we can infer that $\mathtt{Enc}\,C'\,(\sigma\,\overline{e}) \equiv v$. For each encoding, the approximation can be derived by direct calculation, using the minimality of the encoding of rules.

Lemma 17. *Except for* $\mathtt{Lin}_v$, *our encodings satisfy*

$$\mathtt{Enc}\,R\,f \leqslant \mathtt{Enc}\,C'$$

for any rule $R = [C \to C']$ *and closed* f.

Proof. For each of the encodings, except `Lin`, if $W' = D_1 C' D_2$, then

$$\mathtt{Enc}\,W'\overline{\mathtt{x}}\,\overline{\mathtt{y}}'\overline{\mathtt{z}}\mathtt{i}'\mathtt{j}' \leqslant \mathtt{Enc}\,C'\overline{\mathtt{y}}'\mathtt{i}'\mathtt{j}'.$$

That $\mathtt{Enc}\,R \leqslant \lambda\mathtt{f}\,.\,\mathtt{Enc}\,C'$ follows from the minimality condition defining $\mathtt{Enc}\,R$. $\square$

Lemma 18. *If* $R = [C \to C']$ *is a rule*, f *is closed*, *and* `tt` *occurs twice or more in* $\overline{e}'$, *then*

$$\mathtt{LIN}_v\,R\,f\,\overline{e}' \equiv \bot.$$

Proof. Let G be a closed term such that for $x_1 \ldots x_{2l'+2} \in \{\mathtt{tt}, \mathtt{ff}\}$:
- $G\,x_1 \ldots x_{2l'+2} \equiv v$ if $x_i = \mathtt{tt}$ for at most one i.
- $G\,x_1 \ldots x_{2l'+2} \equiv \bot$ if $x_i = \mathtt{tt}$ for two or more i.

Then, for any word W' in the form $D_1 C D_2$, we have that

$$\mathtt{LIN}_v\,W'\overline{\mathtt{x}}\,\overline{\mathtt{y}}'\overline{\mathtt{z}}\mathtt{i}'\mathtt{j}' \leqslant G\overline{\mathtt{y}}'\mathtt{i}'\mathtt{j}',$$

and it follows that $\mathtt{Enc}\,R \leqslant \lambda\mathtt{f}\,.\,G$. $\square$

3. Spine reduction

Throughout the remainder of our argument, $t[\mathtt{W}_0, \mathtt{R}_1, \ldots, \mathtt{R}_N, \mathtt{x}_1, \ldots, \mathtt{x}_{2n+2}]$ represents a term satisfying some word W. For notational convenience, when it does not cause confusion, we omit encodings, e.g., writing W_0 and R_i instead of $\mathtt{Enc}\,W_0$ and $\mathtt{Enc}\,R_i$.

In this section, we will show that if a word is satisfied by a term, then the word is also satisfied by a term that is well behaved in the sense below:

Definition 19. The *coccyx* of a term t is the unique spinal sub-term of t that is *not* in the form $\mathtt{R}_i\,(\lambda\overline{\mathtt{y}}\,.\,s)\,\overline{a}$. A term t is said to have *reduced spine* if its coccyx is in the form $\mathtt{W}_0\,\overline{a}$.

Lemma 20. *For any encoding* $\mathtt{Enc}$,

$$R_i(\lambda\overline{\mathtt{y}}\,.\,\mathtt{if}\ W_0\overline{b}\ \mathtt{then}\ c\ \mathtt{else}\ d)\overline{a}\leqslant R_i(\lambda\overline{\mathtt{y}}\,.\,W_0\overline{b})\overline{a}$$

and

$$R_i(\lambda\overline{\mathtt{y}}\,.\,\mathtt{if}\ R_jg\overline{b}\ \mathtt{then}\ c\ \mathtt{else}\ d)\overline{a}\leqslant R_i(\lambda\overline{\mathtt{y}}\,.\,R_jg\overline{b})\overline{a}.$$

Proof. We do the case of $\mathtt{Enc}$ a $\mathtt{tt}$-encoding. For any word W and any $\overline{y}$,

$$W\overline{y}\equiv\mathtt{if}\ W\overline{y}\ \mathtt{then}\ \mathtt{tt}\ \mathtt{else}\ \bot,$$

because $W\,\overline{y}\leqslant\mathtt{tt}$. Thus, if s is in the form $\mathtt{Enc}\,W\,\overline{e}$, then

$$R_i(\lambda\overline{\mathtt{y}}\,.\,s)\leqslant R_i(\lambda\overline{\mathtt{y}}\,.\,\mathtt{if}\ s\ \mathtt{then}\ \mathtt{tt}\ \mathtt{else}\ \bot). \tag{3}$$

Using the minimality condition that defines $\mathtt{Enc}\,R_i$, this implies that (3) holds for any s. Take s to be $\mathtt{if}\ W_0\,\overline{b}\ \mathtt{then}\ c\ \mathtt{else}\ d$. As $W_0\,\overline{b}\leqslant\mathtt{tt}$, we have

$$\mathtt{if}\ s\ \mathtt{then}\ \mathtt{tt}\ \mathtt{else}\ \bot\leqslant W_0\overline{b},$$

which together with (3) gives the first inequality. The second inequality is similar. □

Lemma 21. *Given any encoding, we have*

$$\mathtt{if}\ W_0\overline{a}\ \mathtt{then}\ b\ \mathtt{else}\ c\geqslant W\overline{x}$$

implies $W_0\,\overline{a}\geqslant W\,\overline{x}$, *and*

$$\mathtt{if}\ R_if\overline{a}\ \mathtt{then}\ b\ \mathtt{else}\ c\geqslant W\overline{x}$$

implies $R_i f\,\overline{a}\geqslant W\,\overline{x}$.

Proof. We do the case of a $\mathtt{ff}$-encoding. WLOG, we may assume that the terms involved are closed. If $W\overline{x}\equiv\bot$, there is nothing to do. If $W\overline{x}\not\equiv\bot$, then $W\overline{x}\equiv\mathtt{ff}$. Now,

$$\mathtt{if}\ W_0\,\overline{a}\ \mathtt{then}\ b\ \mathtt{else}\ c\not\equiv\bot,$$

so that also $W_0\,\overline{a}\not\equiv\bot$. But as we have a $\mathtt{ff}$-encoding, this gives $W_0\,\overline{a}\equiv\mathtt{ff}\geqslant W\overline{x}$ as required. The second inequality is similar. □

Lemma 22. *If a term t satisfies a word W, then the coccyx of t is none of the following*: (a) *a type $\mathscr{B}$ variable* $\mathtt{x}$, (b) *a term in the form* $\mathtt{if\ x\ then}\ \ldots$, (c) *one of* $\mathtt{tt}$, $\mathtt{ff}$ *or* $\bot$.

Proof. Suppose otherwise. We shall apply repeated descent (Lemma 16). We use the encoding $\mathtt{Spine}_v$ where $v \in \{\mathtt{tt}, \mathtt{ff}\}$ is different from the coccyx s of t. Let $\sigma\,\mathtt{x} = \bot$ for all type $\mathscr{B}$ variables $\mathtt{x}$. Appropriate descent functions are the constant undefined functions, so that by the repeated descent lemma, we have that $\sigma\, s \equiv v$, which is clearly impossible. □

Proposition 23 (Spine reduction). *If there is a term t satisfying a word W, then there is a term t' with reduced spine also satisfying W.*

Proof. Consider the coccyx r of t. If r is in the form $\mathtt{W}_0\,\overline{b}$ then t has reduced spine. Otherwise, by Lemma 22, r must be in the form $\mathtt{if}\ \mathtt{W}_0\ \ldots$ or in the form $\mathtt{if}\ \mathtt{R}_j\ \ldots$. In these two cases, we can apply Lemma 21 (if $t = r$) or Lemma 20 (if $t \neq r$) to find a smaller term t' also satisfying W. Repeating, we must eventually arrive at a term with reduced spine. □

4. Rib reduction

We have shown that if a word is satisfied by a term, then the word is satisfied by a term with only $\mathtt{W}_0$ and $\mathtt{R}_i$ in head positions on the spine. We now show that any other occurrences of $\mathtt{W}_0$ and the $\mathtt{R}_i$ may be removed.

Definition 24. The set of *rib sub-terms* of a term t with reduced spine is defined as follows:
- The set of rib sub-terms of $\mathtt{W}_0\, b_1 \ldots b_k$ is $\{b_1, \ldots, b_k\}$.
- The set of rib sub-terms of $\mathtt{R}_i\,(\lambda \overline{\mathtt{y}}\,.\,r)\, c_1 \ldots c_j$ is the union of $\{c_1, \ldots, c_j\}$ with the set of rib sub-terms of r.

A term t, with reduced spine, is said to have *reduced ribs* if $\mathtt{W}_0$ and the $\mathtt{R}_i$ have no occurrences in the set of rib sub-terms of t.

We wish to show that our term t satisfying a word W may be replaced by a term that is rib-reduced. Because $\mathtt{Enc}\, W_0\, \overline{a} \leqslant v$ and $\mathtt{Enc} R_i\, f\, \overline{a} \leqslant v$ for a v-encoding $\mathtt{Enc}$, we could replace $\mathtt{W}_0\, \overline{a}$ and $\mathtt{R}_i f\, \overline{a}$ by v in t, except for the fact that this is not well defined, as v depends on the encoding. However, for occurrences that we wish to replace – those other than spinal sub-terms – the symmetry between the $\mathtt{tt}$- and $\mathtt{ff}$-encodings (expressed by the lemma below) comes to our rescue, as what works for $\mathtt{tt}$-encodings will also work for $\mathtt{ff}$-encodings, and in particular, we may assume $v = \mathtt{tt}$.

Lemma 25. *Suppose that r is a term with reduced spine and reduced ribs, and such that* (2) *is satisfied with* $\mathtt{Enc} = \mathtt{Enc}_{\mathtt{tt}}$ *for some* $\mathtt{tt}$*-encoding* $\mathtt{Enc}_{\mathtt{tt}}$. *Then* (2) *is also satisfied with* $\mathtt{Enc} = \mathtt{Enc}_{\mathtt{ff}}$, *the corresponding* $\mathtt{ff}$*-encoding.*

Proof. Let $\phi = \mathtt{xif.x\ then\ ff\ else\ tt} : \mathscr{B} \Rightarrow \mathscr{B}$. We extend ϕ to any type $A \Rightarrow A$ with $A = A_1 \Rightarrow \cdots \Rightarrow A_n \Rightarrow \mathscr{B}$ by composition: $\phi_A = \lambda \mathtt{f}\overline{\mathtt{x}}.\ \phi(\mathtt{f}\overline{\mathtt{x}})$.

Clearly, $\mathtt{Enc}_{\mathtt{ff}}\, W \equiv \phi(\mathtt{Enc}_{\mathtt{tt}}\, W)$ for any word W, and $\mathtt{Enc}_{\mathtt{ff}}\, R(\phi f) \equiv \phi(\mathtt{Enc}_{\mathtt{tt}}\, Rf)$ for any rule R and term f. These equations supply induction steps to show that for any term r with reduced spine and reduced ribs,

$$r[\mathtt{Enc}_{\mathtt{ff}}\, W_0, \mathtt{Enc}_{\mathtt{ff}}\, \overline{R}, \overline{\mathtt{x}}] \equiv \phi(r[\mathtt{Enc}_{\mathtt{tt}}\, W_0, \mathtt{Enc}_{\mathtt{tt}}\, \overline{R}, \overline{\mathtt{x}}])$$

and the result follows. □

Proposition 26 (Rib reduction). *Suppose that t has reduced spine, and satisfies a word W. Then there is a term with reduced spine and reduced ribs, also satisfying W.*

Proof. Form t' by replacing every occurrence in the form $\mathtt{R}_i f\, \overline{a}$ or $\mathtt{W}_0\, \overline{a}$ within a rib sub-term by $\mathtt{tt}$. t' has reduced spine and reduced ribs. For any $\mathtt{tt}$-encoding $\mathtt{Enc}$, we have

$$\mathtt{Enc}\, W_0\, \overline{a} \leqslant \mathtt{tt} \quad \text{and} \quad \mathtt{Enc}\, R_i f\, \overline{a} \leqslant \mathtt{tt}$$

so that clearly

$$t[\mathtt{Enc}\, W_0, \mathtt{Enc}\, \overline{R}, \overline{\mathtt{x}}] \leqslant t'[\mathtt{Enc}\, W_0, \mathtt{Enc}\, \overline{R}, \overline{\mathtt{x}}].$$

It follows that replacing t by t' leaves (2) still satisfied for any of our $\mathtt{tt}$-encodings. The previous lemma now shows that it is still satisfied for the $\mathtt{ff}$-encodings, so that t' satisfies W. □

From now on, we do not need both the $\mathtt{tt}$- and $\mathtt{ff}$-encodings. We shall use only the $\mathtt{tt}$-encodings, and drop the subscripts from the names of encodings.

5. Rib sanity

We show that the rib sub-terms (of a term t satisfying a word W) can be taken to have certain sensible properties. The rib sub-terms, and type $\mathscr{B}$ variables, occurring in a term may be classified according to certain criteria: 'even' and 'odd', 'parameter' and 'control'. Consider a term

$$\mathtt{W}_0\, a_1 \ldots a_{2l}\, a_{2l+1}\, a_{2l+2}.$$

The rib sub-terms a_i are divided as follows:
- a_{2i-1} $(1 \leqslant i \leqslant l+1)$ are *odd* sub-terms.
- a_{2i} $(1 \leqslant i \leqslant l+1)$ are *even* sub-terms.
- $a_1 \ldots a_{2l}$ are *positional* sub-terms.
- a_{2l+1} and a_{2l+2} are *control* sub-terms.

In the term

$$\mathtt{R}_j\,(\lambda \mathtt{y}_1 \ldots \mathtt{y}_{2k+2}\,.\,b)\,a_1 \ldots a_{2l+2}$$

the sub-terms a_i are classified in the same way, as are the variables $\mathtt{y}_i$ (but using k instead of l). If a term t satisfies a word W w.r.t. $\mathtt{x}_1 \ldots \mathtt{x}_{2n+2}$, then the $\mathtt{x}_i$ are again classified in this manner.

For each i, the term a_{2i-1} is called the *odd partner* of a_{2i} and a_{2i} is called the *even partner* of a_{2i-1}. The partners of variables are defined similarly.

Consider again the condition defining the encoding of rules, and allowing us to encode derivations:

$$\mathtt{Enc}\,W'\overline{\mathtt{x}}\,\overline{\mathtt{y}}'\overline{\mathtt{z}}\,\mathtt{i}'\,\mathtt{j}' \leqslant \mathtt{Enc}\,R\,(\lambda\overline{\mathtt{y}}\,\mathtt{i}\,\mathtt{j}\,.\,\mathtt{Enc}\,W\overline{\mathtt{x}}\,\overline{\mathtt{y}}\,\overline{\mathtt{z}}\,\mathtt{i}\,\mathtt{j})\overline{y}'\mathtt{i}'\mathtt{j}'.$$

Note that only even variables are used in even rib sub-terms, odd variables in odd rib sub-terms, positional variables in positional rib sub-terms, and control variables in control rib sub-terms. By considering encodings that check these four properties, we show that our term t satisfies them also.

Lemma 27 (Local liveness). *Let $e : \mathscr{B}$ be a normal term whose free variables are all of type $\mathscr{B}$. Then there is normal $e' \equiv e$ such that for any variable $\mathtt{x}_0$ occurring in e', there is a substitution σ with the following properties*:

- $\sigma\,\mathtt{x} \in \{\mathtt{tt}, \mathtt{ff}\}$ *for type $\mathscr{B}$ variables other than $\mathtt{x}_0$.*
- $\sigma\,\mathtt{x}_0 = \bot$, *and*
- $\sigma\,e' \equiv \bot$.

Proof. First note that we have the equivalence

$$\mathtt{if\ x\ then}\ b[\mathtt{x}]\ \mathtt{else}\ c[\mathtt{x}] \equiv \mathtt{if\ x\ then}\ b[\mathtt{tt}]\ \mathtt{else}\ c[\mathtt{ff}].$$

We turn this into a reduction by letting the LHS above reduce to the RHS. e may be reduced to a term e' that is normal w.r.t. both this reduction as well as the reductions of Definition 1.

We construct a substitution σ with the required properties, by induction over e'. If e' is $\mathtt{x}_0$, or in the form $\mathtt{if}\ \mathtt{x}_0\ \mathtt{then}\ b\ \mathtt{else}\ c$, then any σ with $\sigma\,\mathtt{x}_0 = \bot$ has $\sigma\,e' \equiv \bot$, as required. If e' is $\mathtt{if\ y\ then}\ b\ \mathtt{else}\ c$, with $\mathtt{y}$ a variable other than $\mathtt{x}_0$, then $\mathtt{x}_0$ occurs in either b or c. We do the case of $\mathtt{x}_0$ occurring in b. The induction hypothesis gives a substitution σ such that $\sigma\,b \equiv \bot$, $\sigma\,\mathtt{x}_0 = \bot$, and $\sigma\,\mathtt{x} \in \{\mathtt{tt}, \mathtt{ff}\}$ for other $\mathtt{x}$. Define $\sigma'\mathtt{y} = \mathtt{tt}$, and $\sigma'\mathtt{x} = \sigma\mathtt{x}$ for all other $\mathtt{x}$. As e' is reduced with respect to the reduction above, $\mathtt{y}$ does not occur in b, and $\sigma'\,e' \equiv \sigma'\,b = \sigma\,b \equiv \bot$ as required. $\square$

Lemma 28 (Class separation). *Suppose that a word W is satisfied by some term t that is spine and rib reduced, and has rib sub-terms satisfying the conclusion of the local liveness lemma. Then the rib sub-terms of t respect the classification above, in that each rib sub-term contains only variables of the same class.*

Proof. Suppose that some rib sub-term e contains a variable $\mathtt{x}_0$ of the wrong class. We use one of the four encodings PosOdd, PosEven, ConOdd and ConEven. Choose the one that corresponds to the class of e; *e.g.*, PosOdd if e is positional and odd, the case considered here. We apply repeated descent (Lemma 16). Let σ be a substitution as given by local liveness (Lemma 27). Since $\sigma\,\mathtt{x} \in \{\mathtt{tt}, \mathtt{ff}\}$ for odd positional variables x, we may take the descent functions needed for repeated descent (Lemma 16) to be constant functions. We then obtain a contradiction immediately as we have a spinal sub-term with odd positional parameter e and $\sigma\,e \equiv \bot$. □

Lemma 29 (Parameter preservation). *Suppose that a word W is satisfied by a term t that is rib and spine reduced. Let e be a rib sub-term of t, and let σ be a substitution assigning* tt *(or* ff*) to every variable in e. Then $\sigma\,e \equiv \mathtt{tt}$ (or $\sigma\,e \equiv \mathtt{ff}$).*

Proof. Use repeated descent (Lemma 16) for the encodings Fixtt (or Fixff), with a substitution given by $\sigma\,\mathtt{x} = \mathtt{tt}$ (or $\sigma\,\mathtt{x} = \mathtt{ff}$) for all variables x. Use constant tt (or ff) valued functions for the descent functions. □

Proposition 30 (Parameter simplicity). *Suppose that a word W is satisfied by a term t that is rib and spine reduced, and has rib sub-terms satisfying the conclusion of the local liveness lemma. Every rib sub-term e of t is equivalent to a variable.*

Proof. If there are no variables occurring in e, then e is a constant, but this would contradict the parameter preservation lemma above.

If there is exactly one variable x in e, then the parameter preservation lemma implies that $e \equiv \mathtt{x}$.

Suppose that the rib sub-term e contains two, or more, distinct variables, including u and v. We consider the case where e is an odd sub-term; the case of even e is similar. By class separation (Lemma 28), the variables u and v are also odd.

Let e' be the even partner of e and let $\mathtt{u}'$ and $\mathtt{v}'$ be the even partners of u and v. Let σ_0 be a substitution defined on even variables, with $\sigma_0\,\mathtt{u}' = \mathtt{tt}$ and $\sigma_0\,\mathtt{x}' = \mathtt{ff}$ for all other even $\mathtt{x}'$. Let $\gamma \in \{\mathtt{tt}, \mathtt{ff}\}$ be such that $\sigma_0\,e' \leqslant \gamma$.

If $\gamma = \mathtt{tt}$, then we take a substitution σ_1, as given by the local liveness lemma, such that $\sigma_1\,e \equiv \bot$ and $\sigma_1\,\mathtt{v} = \bot$, but $\sigma_1\,\mathtt{x} \in \{\mathtt{tt}, \mathtt{ff}\}$ for $\mathtt{x} \neq \mathtt{v}$. If $\gamma = \mathtt{ff}$, then we take σ_1 such that $\sigma_1\,e = \bot$, $\sigma_1\,\mathtt{u} = \bot$ and $\sigma_1\,\mathtt{x} \in \{\mathtt{tt}, \mathtt{ff}\}$ for $\mathtt{x} \neq \mathtt{u}$.

Define σ by $\sigma\,\mathtt{x} = \sigma_1\,\mathtt{x}$ for odd x and $\sigma\,\mathtt{x} = \sigma_0\,\mathtt{x}$ for even x. Note that for an odd rib sub-term b, only odd variables occur in b, so $\sigma\,b \equiv \sigma_1\,b$, while if b' is an even sub-term, then $\sigma\,b' \equiv \sigma_0\,b'$.

We now use the encoding OddSimpγ (either OddSimptt or OddSimpff; if e were even, then we would use EvenSimpγ). Note that σ is chosen so that $\sigma\,\mathtt{x}_{2i} \in \{\mathtt{tt}, \mathtt{ff}\}$, and if $\sigma\,\mathtt{x}_{2i} = \gamma$ then $\sigma\,\mathtt{x}_{2i-1} \in \{\mathtt{tt}, \mathtt{ff}\}$ also. We can now apply repeated descent (Lemma 16) with constant descent functions. But as $\sigma\,e' \equiv \gamma$ and $\sigma\,e \equiv \bot$, we have $\sigma\,s \equiv \bot$ for the spinal sub-term s containing e, which gives a contradiction. □

We summarise our progress so far:

Corollary 31. *Suppose that a word W is satisfied by some term. Then W is satisfied by a term that*
- *is spine-reduced,*
- *is rib-reduced,*
- *each rib sub-term of a given class is a variable of that class.*

6. Spine straightening

Suppose that, for some v-encoding $\mathtt{Enc}$ and some $\overline{z}_1 \overline{y} \overline{z}_2 \, \alpha \, \beta$,

$$R_i \, (\lambda \overline{\mathtt{y}}' \, \mathtt{i}' \, \mathtt{j}' \, . \, W_0 \overline{z}_1 \overline{\mathtt{y}}' \overline{z}_2 \, \mathtt{i}' \, \mathtt{j}') \overline{y} \, \alpha \, \beta \equiv v.$$

Writing s for the LHS, we have that $s \equiv v \geqslant W_0 \overline{z}_1 \overline{y}' \overline{z}_2 \, \alpha' \, \beta'$ for any $\overline{y}' \, \alpha' \, \beta'$. It follows that also

$$R_i \, (\lambda \overline{\mathtt{y}}'' \, \mathtt{i}'' \, \mathtt{j}'' \, . \, s) \overline{y} \, \alpha \, \beta \equiv v,$$

where $\overline{\mathtt{y}}''$, $\mathtt{i}''$ and $\mathtt{j}''$ do not occur in s.

In this fashion we can insert ‘detours’ into a term encoding a word. We must find a way of removing such detours. The first thing to do is to somehow measure where these detours occur within a term. This is the purpose of the control variables.

Definition 32. A term t is called *chain-reduced*, if for every spinal sub-term in the form

$$\mathtt{R}_i \, (\lambda \overline{\mathtt{y}} \, \mathtt{i} \, \mathtt{j} . f \overline{b} \, \alpha \, \beta) \overline{a},$$

we have that $\beta = \mathtt{j}$.

Proposition 33 (Chain reduction). *If a word W is satisfied by some term, then W is satisfied by some term that is in the form given by Corollary* 31, *and that is also chain-reduced.*

Proof. Take t in the form given by Corollary 31. Suppose that t is not chain-reduced. We show how to find a smaller term also satisfying W. Take the inner-most spinal sub-term of t that is not chain reduced:

$$\mathtt{R}_i \, (\lambda \overline{\mathtt{y}} \, \mathtt{i} \, \mathtt{j} . f \overline{b} \, \alpha \, \beta) \overline{a}. \tag{4}$$

Note that $\mathtt{j}$ cannot occur in $f \overline{b} \, \alpha \, \beta$, because if it did, then it would have to occur in an even control position (by class separation, Lemma 28) other than β, and considering a sub-term containing this, we would obtain a smaller sub-term that is not chain-reduced. We now show also that none of the variables $\overline{\mathtt{y}} \, \mathtt{i}$ occur in $f \overline{b} \, \alpha \, \beta$ either.

We apply repeated descent (Lemma 16) with the encoding $\mathtt{Chain}_{\mathtt{tt}}$, and the substitution defined by $\sigma\,\mathtt{i} = \bot$, $\sigma\,\mathtt{y}_k = \bot$, $\sigma\,\mathtt{j} = \mathtt{ff}$, and $\sigma\,\mathtt{x} = \mathtt{tt}$ for all other $\mathtt{x} : \mathscr{B}$. Descent functions for the sub-term (4) are given by constant functions, while for other sub-terms in the form

$$\mathtt{R}_j\,(\lambda \mathtt{z}_1 \ldots \mathtt{z}_{2l+2}\,.\,h)\,c_1 \ldots c_{2l'+2}$$

the descent functions are given by $g_p\,\overline{u} \equiv \mathtt{tt}$ (for $1 \leqslant p \leqslant 2l+1$) and $g_{2l+2}\,u_1 \ldots u_{2l'+2} \equiv u_{2l'+2}$. Since $\mathtt{j}$ does not occur in t (other than in the indicated λ-binding) we have that $\sigma\,d \equiv \mathtt{tt}$ for each even control rib sub-term d. Now, if one of the $\mathtt{y}_k$ or $\mathtt{i}$ occurs in t, we have that $\sigma\,e \equiv \bot$ for some positional or odd control rib sub-term e. Using the definition of the encoding $\mathtt{Chain}$ gives a contradiction.

We have established that none of the variables $\overline{\mathtt{y}}\ \mathtt{i}\ \mathtt{j}$ occur in $f\,\overline{b}\alpha\beta$. By constant descent (Corollary 14), we have that

$$R_i\,(\lambda \overline{\mathtt{y}}\,\mathtt{i}\,\mathtt{j}.f\,\overline{b}\alpha\beta)\overline{a} \leqslant f\,\overline{b}\alpha\beta$$

for any encoding, so that we may make the required reduction of the term t, by replacing the LHS above by the RHS. □

7. Linearity

The chain reduction proposition establishes that our term t may be taken to have unique occurrences of even parameter variables. We use the encoding $\mathtt{Lin}$ to establish that other variables have unique occurrences. This ensures that sub-terms have the correct context in our final induction that proves the faithfulness of the encodings.

Lemma 34. *Let $t[\mathtt{W}_0, \overline{\mathtt{R}}, \mathtt{x}_1 \ldots \mathtt{x}_{2n+2}]$ satisfy a word W, and have all the previous reductions applied. Then each $\mathtt{x}_i$ occurs in t.*

Proof. Suppose otherwise, that $\mathtt{x}_i$ does not occur in t. We use repeated descent (Lemma 16), with the encoding $\mathtt{Lin}_{\mathtt{tt}}$. Take a substitution σ assigning $\mathtt{tt}$ to $\mathtt{x}_i$ and $\mathtt{ff}$ to all other variables. As $\mathtt{x}_i$ does not occur in t, we have $\sigma\,a \equiv \mathtt{ff}$ for every rib sub-term a of t, and descent functions can be taken to be those given in the proof of descent existence (Lemma 13). Applying repeated descent, we infer that $\sigma\,s \equiv \mathtt{tt}$, where s is the coccyx of t. But s is in the form $W_0\overline{\mathtt{z}}$, where $\sigma\,\mathtt{z}_j = \mathtt{ff}$ for each $\mathtt{z}_j$, which makes $\sigma\,s \equiv \bot$, a contradiction. □

Proposition 35 (Linearity). *Let $t[\mathtt{W}_0, \overline{\mathtt{R}}, \mathtt{x}_1 \ldots \mathtt{x}_{2n+2}]$ satisfy a word W, and have all the previous reductions applied. Then each $\mathtt{x}_j$ occurs in t exactly once.*

Proof. Suppose that $\mathtt{x}_j$ occurs more than once in t. Note that as t is chain reduced, class separation (Lemma 28) implies that $j \leqslant 2n+1$. We consider the case when t has occurrences of $\mathtt{x}_j$ at different levels within t, i.e., there is a spinal sub-term s in the form

$$\mathtt{R}_i\,(\lambda\overline{\mathtt{y}}\,.\,r)\,\overline{b},$$

where $\mathtt{x}_j$ is one of the $\overline{b}$ and also occurs in r. (The other case is easier.) Take s to be the largest such sub-term. We shall apply repeated descent (Lemma 16) with the encoding $\mathtt{Lin}_{\mathtt{tt}}$ to derive a contradiction. Define a substitution σ as follows: $\sigma\,\mathtt{x}_j = \mathtt{tt}$, $\sigma\,\mathtt{i} = \mathtt{tt}$ for any even control variable $\mathtt{i}$ whose *binder* occurs *within* s, and $\sigma\,\mathtt{y} = \mathtt{ff}$ for any other variable $\mathtt{y}$. Appropriate descent functions are as given in the proof of descent existence (Lemma 13).

Let s' be a spinal sub-term of r with $\mathtt{x}_j$ one of its outermost rib sub-terms. By repeated descent, $\sigma\,s' \equiv \mathtt{tt}$. s' is in one of the forms $\mathtt{R}_{i'}\,f\,\overline{c}\,\mathtt{j}$ or $\mathtt{W}_0\,\overline{c}\,\mathtt{j}$, where one of the $\overline{c}$ is $\mathtt{x}_j$. As t is chain reduced, the binder of $\mathtt{j}$ occurs within s, so $\sigma\,\mathtt{j} = \mathtt{tt}$. As also $\sigma\,\mathtt{x}_j = \mathtt{tt}$, by Lemma 18, this gives $\sigma\,s' \equiv \bot$, a contradiction. $\square$

8. Faithfulness

We can now show that our encoding is faithful, from which the undecidability of $\equiv$ and $\leqslant$ follows immediately. We do this by induction over a term satisfying a word.

The lemma below gives the main calculation of the induction step, once we have sorted out what words and rules are involved, and what variables occur where. By inspecting the proof of Theorem 37, we in fact only need this result for the encodings `Word`, `Lin`, `PosCh` and `PosEq`.

Lemma 36 (Descent completeness). *Let* `Enc` *be one of our* v*-encodings. Suppose that* $R = [C \to C']$, $W = D_1CD_2$ *and* $W' = D_1C'D_2$. *Let* l, l', k_1 *and* k_2 *be the lengths of* C, C', D_1 *and* D_2. *Suppose that*

$$\mathtt{Enc}\,W\,\overline{x}\,\overline{y}\,\overline{z}\,\alpha\,\beta \equiv v,$$

where $\bar{x}$, $\overline{y}$ *and* $\bar{z}$ *have lengths* $2k_1$, $2l$ *and* $2k_2$. *Then there are* $\overline{y}'$, α' *and* β' *and descent functions* $\overline{g}$ *for* $\mathtt{Enc}\,R$ *such that*

$$y_i \equiv g_i\,\overline{y}'\,\alpha'\,\beta' \quad (1 \leqslant i \leqslant 2l), \qquad \alpha \equiv g_{2l+1}\,\overline{y}'\,\alpha'\,\beta', \qquad \beta \equiv g_{2l+2}\,\overline{y}'\,\alpha'\,\beta'$$

and

$$\mathtt{Enc}\,W'\,\overline{x}\,\overline{y}'\,\overline{z}\,\alpha'\,\beta' \equiv v.$$

Proof. In most cases, we can simply take the g_i to be the y_i-, α- and β-valued constant functions, and then choose satisfactory y', α' and β' (e.g., for `Word`, take y'_{2i} and y'_{2i+1} to be the ith letter of C'). For `PosCh`, set

$$g_1\,\overline{y}'\,\alpha'\,\beta' \equiv y'_1 \quad \text{and} \quad g_{2l}\,\overline{y}'\,\alpha'\,\beta' \equiv y_{2l'}$$

with the other g_i constant functions, and take $y'_1 = y_1$, $y'_{2l'} = y_{2l}$, $y'_{2i} = y'_{2i+1} = \mathtt{tt}$. The cases of Chain and Lin are left to the reader. □

Theorem 37. *Our encodings are faithful: if a term t satisfies a word W, then the word W is semi-Thue derivable from W_0 using the rules $\overline{R}$. Thus the observational pre-order and the observational equivalence are undecidable.*

Proof. We assume that all the reductions given in the preceding sections have been applied to $t[\mathtt{W}_0, \overline{\mathtt{R}}, \overline{\mathtt{x}}, \mathtt{i}, \mathtt{j}]$. We now proceed via induction on t. For the base case suppose that t has head $\mathtt{W}_0$. By class separation (Lemma 28) and linearity (Proposition 35), t is in the form

$$\mathtt{W}_0\,\overline{a}\,\mathtt{i}\,\mathtt{j}$$

with $\overline{a}$ some permutation of $\mathtt{x}_1 \ldots \mathtt{x}_{2n}$.

Given p, q with $1 \leqslant p$, $q \leqslant n$, let σ be a substitution with $\sigma\,\mathtt{x}_{2q-1} = \sigma\,\mathtt{x}_{2q} = \mathtt{tt}$, and $\sigma\,\mathtt{y} = \mathtt{ff}$ for other type $\mathscr{B}$ variables. By considering the encoding PosEq, we have that $\sigma\,a_{2p-1} \equiv \mathtt{tt}$ iff $\sigma\,a_{2p} \equiv \mathtt{tt}$, and so by class separation (Lemma 28), we have that

$$a_{2p-1} = \mathtt{x}_{2q-1} \quad \text{iff} \quad a_{2p} = \mathtt{x}_{2q}. \tag{5}$$

Given p, q with $1 \leqslant p$, $q < n$, let τ be a substitution with $\tau\,\mathtt{x}_{2q} = \tau\,\mathtt{x}_{2q+1} = \mathtt{tt}$ and $\tau\,\mathtt{y} = \mathtt{ff}$ for other type $\mathscr{B}$ variables. By considering the encoding PosCh, we have that $\tau\,a_{2p} \equiv \mathtt{tt}$ iff $\tau\,a_{2p+1} \equiv \mathtt{tt}$, and so by class separation (Lemma 28), we have that

$$a_{2p} = \mathtt{x}_{2q} \quad \text{iff} \quad a_{2p+1} = \mathtt{x}_{2q+1}. \tag{6}$$

Let $\rho\,\mathtt{x}_1 = \perp$, and $\rho\,\mathtt{y} = \mathtt{tt}$ for other type $\mathscr{B}$ variables. For $1 < p \leqslant n$, by considering the encoding PosCh, we have that $\rho\,a_p \equiv \perp$, so that

$$a_p \neq \mathtt{x}_1. \tag{7}$$

The three conditions (5)–(7) imply that $\overline{a} = \overline{\mathtt{x}}$. Let w_{2p-1} and w_{2p} be the pth letter of W for $1 \leqslant p \leqslant n$. As t satisfies W, we have

$$\mathtt{Word}_{\mathtt{tt}}\,W_0\,\overline{w} \perp \perp \geqslant \mathtt{Word}_{\mathtt{tt}}\,w\,\overline{w} \perp \perp \equiv \mathtt{tt}$$

and so $W = W_0$, which is derivable, as required.

For the induction step, suppose that $t[\mathtt{W}_0, \overline{\mathtt{R}}, \overline{\mathtt{x}}, \mathtt{i}', \mathtt{j}']$ is in the form

$$\mathtt{R}_i\,(\lambda \overline{\mathtt{y}}\,\mathtt{i}\,\mathtt{j}\,.\,s)\,\overline{a}\,\mathtt{i}'\,\mathtt{j}'.$$

with $R_i = [C \to C']$, where C and C' have lengths l and l'. Arguing as in the base case, we have (5), for $1 \leqslant p \leqslant l'$ and $1 \leqslant q \leqslant n$, and (6), for $1 \leqslant p < l'$ and $1 \leqslant q < n$, and (7) for $1 < p \leqslant l'$. These imply that $\overline{\mathtt{x}}$ is in the form $\overline{\mathtt{z}}_1 \overline{a}\,\overline{\mathtt{z}}_2$, with $\overline{\mathtt{z}}_1$ and $\overline{\mathtt{z}}_2$ having even lengths, say $2k_1$ and $2k_2$. By linearity (Proposition 35), the term $s[\mathtt{W}_0, \overline{\mathtt{R}}, \overline{\mathtt{z}}_1, \overline{\mathtt{y}}, \overline{\mathtt{z}}_2, \mathtt{i}, \mathtt{j}]$ has only the indicated free variables.

Let π be a substitution such that $\pi\,\mathtt{x}_{2p-1}$ and $\pi\,\mathtt{x}_{2p}$ are the pth letter of W, for $1 \leqslant p \leqslant n$, and such that $\pi\mathtt{i} = \pi\mathtt{j} = \bot$. Using the encoding $\mathtt{Word}_{\mathtt{tt}}$ and Lemma 17, we have

$$C'\,(\pi\,\overline{\mathtt{y}})\perp\perp \geqslant R_i\,(\lambda\overline{\mathtt{y}}\,\mathtt{i}\,\mathtt{j}\,.s[W_0,\overline{R},\pi\,\overline{\mathtt{z}}_1,\overline{\mathtt{y}},\pi\,\overline{\mathtt{z}}_2,\mathtt{i},\mathtt{j}])\,(\pi\,\overline{a})\perp\perp.$$

The RHS above is $t[W_0,\overline{R},\pi\,\mathtt{x},\bot,\bot]$, which is $\equiv\mathtt{tt}$ as t satisfies W. This shows that W is in the form $D_1C'D_2$, where D_1 and D_2 have lengths k_1 and k_2.

To complete the induction step, it suffices to show that s satisfies D_1CD_2, as by the induction hypothesis, this implies that D_1CD_2, and so also W, are derivable. Let $\mathtt{Enc}$ be one of our v-encodings, and suppose that $\overline{x}\,\overline{y}\,\overline{z}\,\alpha\,\beta$ are such that

$$\mathtt{Enc}(D_1CD_2)\,\overline{x}\,\overline{y}\,\overline{z}\,\alpha\,\beta \equiv v.$$

Let $\overline{y}'$, α' and β' and $\overline{g}$ be as given by descent completeness (Lemma 36). Then $\mathtt{Enc}\,W\,\overline{x}\,\overline{y}'\,\overline{z}\,\alpha'\,\beta' \equiv v$, and as t satisfies W, we have that

$$t[W_0,\overline{R},\overline{x},\overline{y}',\overline{z},\alpha',\beta'] \equiv v.$$

By the properties of the descent functions $\overline{g}$ given by descent completeness, we have also

$$s[W_0,\overline{R},\overline{x},\overline{y},\overline{z},\alpha,\beta] \equiv v.$$

This shows that s satisfies D_1CD_2, as required. □

The proof we have given in fact shows that $\equiv$ is undecidable on fifth-order types – the encodings of words have order two, the encodings of rules have order three, so the variable in the equations of Corollary 6 has order four, and the terms constructed in the proof of Lemma 5 have order five. (We use $\mathrm{order}(\mathscr{B}) = 1$. Note that both 0 and 1 are used in the literature.)

This is optimal, in that the results of [16] show that complete sets of equivalence classes at types of order three may be calculated, and thus equivalence at order four can be effectively tested. As concerns the lengths of the types involved, these are inherited from the semi-Thue system used; see [8] for undecidable systems that are efficient in this regard.

The decidability results of Padovani [12] and the author [7] show that our result requires the whole language of PCF_2, in that obvious restrictions yield languages with nice decidability properties.

Finally, we note that our result gives a new proof of the earlier result [6] of the author that typed λ-definability is undecidable. A detailed proof of the implication can be found in [5]. The proof consists of noting that the fully abstract model is given by taking the extensional collapse of the elements of the full type hierarchy over $\{\mathtt{tt},\mathtt{ff},\bot\}$ that are definable relative to $\mathtt{tt}$, $\mathtt{ff}$, $\bot$ and $\mathtt{if}$. If the definability problem in that full type hierarchy were decidable, then the construction would give an effective presentation of the fully abstract model of PCF_2. This proof applies to the full type hierarchy with three (or more) elements at ground type, as opposed to seven (or more) in the original proof, and is thus a slight improvement. Decidability of λ-definability in the full type hierarchy with two elements at ground type appears to still be an open problem, although this is does not seem to be a matter of any importance.

9. Conclusion

We have shown that the observational equivalence of PCF_2 is undecidable. Thus this decidability question is useless as a measure of the success of a solution to the full abstraction problem of PCF, and if one were to take showing such decidability as a necessary condition for solving the full abstraction problem, then the full abstraction problem would have no solution.

Following this, there are two questions that should be considered. One question is 'what mathematical results should one expect to be implied by a good solution to the full abstraction problem for PCF?'. But one should maybe first ask 'is the previous question a useful one to ask?'

The techniques invented for attacking the full abstraction problem, especially game semantics, have turned out to be useful in the semantics of a variety of different computational languages. In particular, there have been successes in modelling a variety of non-functional constructs (e.g., state and side-effects in [2]), which appear to have been outside of the reach of more traditional denotational semantics, such as domains.

These results indicate that it is the techniques, rather than results about PCF, that are proving important. For this reason, it would seem that requiring a full abstraction result for PCF to imply some specific mathematical result about PCF and its models, is not a particularly useful goal.

The fully abstract term model constructed by Milner is small, in the sense that if $\mathscr{C}$ is any category giving a fully abstract model of PCF, then the term model is equivalent to some full subcategory of $\mathscr{C}$, with the embedding preserving the interpretation of PCF. Constructions such as game models, on the other hand, can give large models, e.g., being proper classes in the set-theoretic sense.

This suggests that one could look for a solution to the full abstraction problem that is a maximum solution, in the sense of having any other fully abstract model equivalent to a full subcategory (with the embedding appropriately structure preserving). Such a result would give (at least in theory) a uniform and general method for deriving conservativity results such as those of [15]. A construction of such a maximal model is probably given abstractly as something like the category of sheaves over the term model, but it seems unclear whether or not models such as the game models provide a maximum model.

Acknowledgements

The author is indebted to referees for comments that greatly clarified the detail and structure of the arguments here.

Appendix

We give below the details of the encodings used. We only define terms up to $\equiv$. When presenting a term of type $\mathscr{B} \Rightarrow \cdots \Rightarrow \mathscr{B} \Rightarrow \mathscr{B}$, we only specify on which argument values the function takes the value $\mathtt{tt}$ or $\mathtt{ff}$, the function is assumed to be $\perp$ everywhere else. The verification that there are actually terms satisfying the given specifications are straightforward and omitted.

We state the encodings applied to a word W of length n. All the encodings come in pairs, a $\mathtt{tt}$-encoding $\mathtt{Enc}_{\mathtt{tt}}$ and a $\mathtt{ff}$-encoding $\mathtt{Enc}_{\mathtt{ff}}$, such that $\mathtt{Enc}_{\mathtt{tt}}\, W\, x_1 \dots x_{2n+2} \equiv \mathtt{tt}$ if and only if $\mathtt{Enc}_{\mathtt{ff}}\, W\, x_1 \dots x_{2n+2} \equiv \mathtt{ff}$.

1. If W is the word $w_1 \dots w_n$, then

$$\mathtt{Word}_v\, W\, w_1\, w_1 \dots w_n\, w_n\, \alpha\, \alpha' \equiv v.$$

2. For any $x_1 \dots x_{2n+2}$,

$$\mathtt{Spine}_v\, W\, x_1 \dots x_{2n+2} \equiv v.$$

3. If $x_1 \dots x_n \in \{\mathtt{tt}, \mathtt{ff}\}$, then

$$\mathtt{PosOdd}_v\, W\, x_1\, x_1' \dots x_n\, x_n'\, \alpha\, \alpha' \equiv v.$$

4. If $x_1' \dots x_n' \in \{\mathtt{tt}, \mathtt{ff}\}$, then

$$\mathtt{PosEven}_v\, W\, x_1\, x_1' \dots x_n\, x_n'\, \alpha\, \alpha' \equiv v.$$

5. If $\alpha \in \{\mathtt{tt}, \mathtt{ff}\}$, then

$$\mathtt{ConOdd}_v\, W\, x_1\, x_1' \dots x_n\, x_n'\, \alpha\, \alpha' \equiv v.$$

6. If $\alpha' \in \{\mathtt{tt}, \mathtt{ff}\}$, then

$$\mathtt{ConEven}_v\, W\, x_1\, x_1' \dots x_n\, x_n'\, \alpha\, \alpha' \equiv v.$$

7. If for $1 \leqslant i \leqslant n+1$, either $x_i = \mathtt{tt}$ and $x_i' \in \{\mathtt{tt}, \mathtt{ff}\}$, or $x_i = \mathtt{ff}$, then

$$\mathtt{EvenSimptt}_v\, W\, x_1\, x_1' \dots x_{n+1}\, x_{n+1}' \equiv v.$$

8. If for $1 \leqslant i \leqslant n+1$, either $x_i = \mathtt{ff}$ and $x_i' \in \{\mathtt{tt}, \mathtt{ff}\}$, or $x_i = \mathtt{tt}$, then

$$\mathtt{EvenSimpff}_v\, W\, x_1\, x_1' \dots x_{n+1}\, x_{n+1}' \equiv v.$$

9. If for $1 \leqslant i \leqslant n+1$, either $x_i' = \mathtt{tt}$ and $x_i \in \{\mathtt{tt}, \mathtt{ff}\}$, or $x_i' = \mathtt{ff}$, then

$$\mathtt{OddSimptt}_v\, W\, x_1\, x_1' \dots x_{n+1}\, x_{n+1}' \equiv v.$$

10. If for $1 \leqslant i \leqslant n+1$, either $x_i' = \mathtt{ff}$ and $x_i \in \{\mathtt{tt}, \mathtt{ff}\}$, or $x_i' = \mathtt{tt}$, then

$$\mathtt{OddSimpff}_v\, W\, x_1\, x_1' \dots x_{n+1}\, x_{n+1}' \equiv v.$$

11. If $x_1 = \cdots = x_{2n+2} = \mathtt{tt}$ then

$$\mathtt{Fixtt}_v\, W\, x_1 \ldots x_{2n+2} \equiv v.$$

12. If $x_1 = \cdots = x_{2n+2} = \mathtt{ff}$ then

$$\mathtt{Fixff}_v\, W\, x_1 \ldots x_{2n+2} \equiv v.$$

13. If either $\alpha' = \mathtt{tt}$ and $x_1 \ldots x_{2n}\, \alpha \in \{\mathtt{tt}, \mathtt{ff}\}$, or $\alpha' = \mathtt{ff}$, then

$$\mathtt{Chain}_v\, W\, x_1 \ldots x_{2n}\, \alpha\, \alpha' \equiv v.$$

14. If $x_i = x_i' \in \{\mathtt{tt}, \mathtt{ff}\}$ for $1 \leqslant i \leqslant n$, then

$$\mathtt{PosEq}_v\, W\, x_1\, x_1' \ldots x_n\, x_n'\, \alpha\, \alpha' \equiv v.$$

15. If $x_{2i} = x_{2i+1} \in \{\mathtt{tt}, \mathtt{ff}\}$ for $1 \leqslant i \leqslant n-1$ then

$$\mathtt{PosCh}_v\, W\, x_1 \ldots x_{2n}\, \alpha\, \alpha' \equiv v.$$

16. For $1 \leqslant i \leqslant 2n+2$,

$$\mathtt{Lin}_v\, W\, \underbrace{\mathtt{ff} \ldots \mathtt{ff}}_{i-1}\, \mathtt{tt}\, \underbrace{\mathtt{ff} \ldots \mathtt{ff}}_{2n+2-i} \equiv v,$$

References

[1] S. Abramsky, R. Jagadeesan, P. Malacaria, Full abstraction for PCF, to appear.

[2] S. Abramsky, G. McCusker, A fully abstract game semantics for idealized algol with active expressions, in: Proc. 1996 Workshop on Linear Logic, Electronic Notes in Theoretical Computer Science, vol. 3, Elsevier, Amsterdam, 1996.

[3] Val Breazu-Tannen, J. Gallier, Polymorphic rewriting conserves algebraic confluence, Inform. and Comput. 114 (1994) 1–29.

[4] J.M.E. Hyland, Luke Ong, On full abstraction for PCF, to appear.

[5] A. Jung, A. Stoughton, Studying the fully abstract model of PCF within its continuous function model, in: M. Bezem, J.F. Groote (Eds.), Typed Lambda Calculi and Applications, Lecture Notes in Computer Science, vol. 664, Springer, Berlin, 1993, pp. 230–244.

[6] R. Loader, Lambda Definability is Undecidable, in: A. Anderson, M. Zeleny (Eds.), Church Memorial Volume, Kluwer Academic Press, Dordrecht, to appear.

[7] R. Loader, Unary PCF is decidable, Theoret. Comput. Sci., to appear.

[8] Y. Matiyasevich, Word problems for Thue systems with a few relations, in: H. Comon, J.-P. Jouannaud (Eds.), Term Rewriting, Lecture Notes in Computer Science, vol 909, Springer, Berlin, 1993, pp. 39–53.

[9] R. Milner, Fully abstract models of typed Lambda calculi, Theoret. Comput. Sci. 4 (1977) 1–22.

[10] H. Nickau, Hereditarily sequential functionals: a game-theoretic approach to sequentiality, Doctoral Dissertation. Shaker Verlag, 1996.

[11] P. O'Hearn, J. Riecke, Kripke logical relations and PCF, Inform. and Comput. 120 (1995) 107–116.

[12] V. Padovani, Decidability of all minimal models, in: M. Coppo et al. (Eds.), Proc. BRA Types Workshop, Torino, June 1995, to appear.

[13] G. Plotkin, λ-definability and logical relations, Memorandum SAI-RM-4, School of Artificial Intelligence, University of Edinburgh, 1993.

[14] G. Plotkin, LCF considered as a programming language, Theoret. Comput. Sci. 5 (1977) 223–255.

[15] J. Riecke, R. Subrahmanyam, Extensions to type systems can preserve operational equivalences, in: M. Hagiya, J. Mitchell (Eds.), Proc. 1994 Internat. Symp on Theoretical Aspects of Computer Software, Lecture Notes in Computer Science, vol. 789, Springer, Berlin, pp. 76–95.
[16] K. Sieber, Reasoning about sequential functions via logical relations, in: M. Fourman, P. Johnstone, A. Pitts (Eds.), Applications of Categories in Computer Science, LMS Lecture Note Series, vol. 177, Cambridge, 1992, pp. 258–269.
[17] R. Statman, Logical relations and the typed λ-calculus, Inform. and Control 65 (1985) 85–97.