BRICS RS-97-24 S. B. Lassen: Relational Reasoning about Contexts

BRICS

Basic Research in Computer Science

Relational Reasoning about Contexts

Søren B. Lassen

BRICS Report Series RS-97-24

ISSN 0909-0878 September 1997

See back inner page for a list of recent BRICS Report Series publications. Copies may be obtained by contacting:

BRICS
Department of Computer Science
University of Aarhus
Ny Munkegade, building 540
DK–8000 Aarhus C
Denmark

Telephone: +45 8942 3360
Telefax: +45 8942 3255
Internet: BRICS@brics.dk

BRICS publications are in general accessible through the World Wide Web and anonymous FTP through these URLs:

`http://www.brics.dk`
`ftp://ftp.brics.dk`
This document in subdirectory `RS/97/24/`

Relational Reasoning about Contexts*

S. B. Lassen
BRICS†, Department of Computer Science, University of Aarhus

1 Introduction

The syntactic nature of operational reasoning requires techniques to deal with term contexts, especially for reasoning about recursion. In this paper we study applicative bisimulation and a variant of Sands' improvement theory for a small call-by-value functional language. We explore an indirect, relational approach for reasoning about contexts. It is inspired by Howe's precise method for proving congruence of simulation orderings and by Pitts' extension thereof for proving applicative bisimulation up to context. We illustrate this approach with proofs of the unwinding theorem and syntactic continuity and, more importantly, we establish analogues of Sangiorgi's bisimulation up to context for applicative bisimulation and for improvement. Using these powerful bisimulation up to context techniques, we give concise operational proofs of recursion induction, the improvement theorem, and syntactic minimal invariance. Previous operational proofs of these results involve complex, explicit reasoning about contexts.

Related work

Applicative bisimulation (Abramsky 1990) is an operational theory for higher-order languages, inspired by bisimulation theories for concurrency (Park 1981; Milner 1989). It excels in reasoning about infinite data structures. These exist in every higher-order language but are particularly relevant in lazy functional languages (Gordon 1995; Pitts 1997) and functional object-oriented languages (Gordon and Rees 1996). But applicative bisimulation is not very helpful for reasoning about recursive control structures. There are more 'intensional' operational theories (Talcott 1997; Sands 1997b) which address recursion effectively by counting computation steps. But even they are of limited use for proving results such as the validity of the fundamental induction rules for recursion: recursion induction (also known as Park induction), syntactic continuity (ω induction), syntactic minimal invariance (syntactic projections), and the improvement theorem. Existing operational proofs are complex and involve explicit reasoning about term contexts.

Intuitively, a context is a term containing a hole, that may be filled by another term. This is an evocative idea, but for formal arguments contexts are difficult

*To appear in Gordon and Pitts (1997).

†Basic Research in Computer Science, Centre of the Danish National Research Foundation.

to work with, both technically and notationally. For this reason, Howe deals only indirectly with contexts in his influential congruence proof for applicative bisimilarity (Howe 1989; Howe 1996). Instead the proof is ‘relational’: a larger relation which is closed under contexts is constructed and is shown to coincide with applicative bisimilarity by bisimulation and induction on the evaluation relation. This relational approach yields a formally and notationally very precise proof. Moreover, Howe’s congruence proof applies to many different typed and untyped higher-order languages and operational orderings; see, e.g., Sands (1991, Ong (1992, Ferreira, Hennessy, and Jeffrey (1995, Lassen (1997, Gordon (1997).

Pitts (1995) extends Howe’s congruence proof for applicative bisimilarity to also establish an up to context rule for applicative bisimulation. The proof is also ‘relational’ and illustrates the versatility of Howe’s implicit, relational approach to reasoning about term contexts. Specifically it shows how to use this proof method to establish applicative bisimulation up to context results. (We present the proof in Section 5.2.) The results we present in this paper stem from the study of this work.

Sangiorgi’s bisimulation up to context is a powerful refined bisimulation proof rule for process calculi (Sangiorgi and Milner 1992; Sangiorgi 1994). Bisimulation up to context allows you to disregard a common term context when relating terms in bisimulation proofs. Unfortunately, his correctness proofs do not carry over to applicative bisimulation for higher-order languages. Gordon (1995) and Sands (1997b) present restricted applicative bisimulation up to context rules. They demonstrate the power of this approach to produce concise proofs of equivalences which are difficult to derive by other operational methods. Both Sangiorgi (1996, 1995) and Sands couple bisimulation up to context with efficiency preorders, called ‘expansion’ and ‘improvement’, respectively. As suggested by Pitts (1995), we also introduce an improvement preorder. The problem which we address in this fashion leads us to adopt an improvement theory based on a different cost measure than that of Sands (1997b).

Overview

Section 2 defines the syntax and operational semantics of the untyped, functional ML fragment which we study below. Section 3 introduces an algebra of relations on terms. This is essential for the calculations with relations in later sections. A substantial example is the proof of the unwinding theorem in Section 4. Applicative (bi)simulation is defined in Section 5. Preliminary applicative simulation up to context results are established and applicative bisimilarity is shown to be a congruence by Howe’s and Pitts’ techniques. A deficiency of applicative simulation up to context is discovered which leads us to introduce an improvement preorder in Section 6. Improvement enjoys a strong up to context rule from which congruence and the improvement theorem follow. Section 7 uses improvement to strengthen the applicative simulation up to context rule from Section 5. Finally, Section 8 concludes. An appendix contains proofs from Sections 6 and 7.

2 A functional ML fragment

We operate with a small call-by-value functional language with lists, an untyped fragment of ML (Milner, Tofte, and Harper 1990).

Syntax

Let f, g, x, y, z range over an infinite set of variables. The syntax of expressions is:

$$\begin{array}{llll} (\mathit{Exp}) & d, e & ::= & x \mid \texttt{fn}\ x\ \texttt{=>}\ e \mid \texttt{nil} \mid e_1\ \texttt{::}\ e_2 \mid e_1\, e_2 \\ & & \mid & \texttt{let fun}\ f\, x = d\ \texttt{in}\ e\ \texttt{end} \mid \texttt{let val}\ x = d\ \texttt{in}\ e\ \texttt{end} \\ & & \mid & (\texttt{case}\ d\ \texttt{of nil =>}\ e_1 \mid x_1\ \texttt{::}\ x_2\ \texttt{=>}\ e_2 \mid f\ \texttt{=>}\ e_3). \end{array}$$

Expressions are identified up to α-renaming of bound variables.

In `let val` $x = d$ `in` e `end` and `fn` x `=>` e, x is bound in e.

In `let fun` $f\,x = d$ `in` e `end`, f and x are bound in d, and f is bound in e.

In `case` d `of nil =>` e_1 `|` x_1 `::` x_2 `=>` e_2 `|` f `=>` e_3, x_1 and x_2 are bound in e_2, and f is bound in e_3.

Terms are parsed as in ML. The scope of `fn` and `case` extends as far to the right as possible. Application associates to the left and has higher precedence than `::` which associates to the right. For instance, the term `fn` x `=>` x `::` y `::` $x\,y\,z$ parses as `fn` x `=>` $(x\ \texttt{::}\ (y\ \texttt{::}\ ((x\,y)\,z)))$.

The set of values is given by the grammar:

$$(\mathit{Val}) \qquad u, v, w \quad ::= \quad x \mid \texttt{fn}\ x\ \texttt{=>}\ e \mid \texttt{nil} \mid v_1\ \texttt{::}\ v_2.$$

Let $e\{\vec{v}/\vec{x}\} = e\{v_1/x_1, \ldots, v_n/x_n\}$ be the result of simultaneous, capture free substitution of values $\vec{v} = v_1 \ldots v_n$ for free occurrences of $\vec{x} = x_1 \ldots x_n$ in e. (See Stoughton (1988) for a precise definition of simultaneous substitution.) By $\vec{x}$ we always mean an ordered list of pairwise distinct variables. We write $x \in \vec{x}$ to mean variable x occurs in $\vec{x}$.

Let $\mathit{Exp}_{\vec{x}}$ and $\mathit{Val}_{\vec{x}}$ be the set of expressions and values, respectively, with free variables contained in $\vec{x}$. Notice $\mathit{Val}_{\vec{x}} \subseteq \mathit{Exp}_{\vec{x}}$. We call expressions $p, q \in \mathit{Exp}_{\emptyset}$ closed.

A closed value is either the empty list `nil`, 'cons' of two closed values v_1 `::` v_2, or a function `fn` x `=>` e with $e \in \mathit{Exp}_x$. The case construct has three corresponding branches. This allows both decomposition of lists and dynamic dispatch on the 'type' of values. (The latter would not be well-typed in ML but is common in untyped languages, e.g., Scheme (Clinger and Rees (editors) 1991) has a `proc?` predicate that tells whether a value is a closure; this feature is necessary for the formulation of syntactic minimal invariance in Proposition 11 but otherwise our results are unaffected by the exact choice of language constructs for accessing values—as long as application is the only means of ‘destructing’ functions.)

We take $\texttt{let val}\ x = d\ \texttt{in}\ e\ \texttt{end}$ as a language primitive instead of encoding it as $(\texttt{fn}\ x\ \texttt{=>}\ e)\ d$, because the encoding introduces a function application step. This difference affects the improvement theory of Section 6 and will be important later in the proof of Proposition 11.

We define Ω to be a divergent expression:

$$\Omega \stackrel{\text{def}}{=} \texttt{let fun}\ f\ x = f\ f\ \texttt{in}\ f\ f\ \texttt{end}.$$

We write $\texttt{rec}\ f\ x\ \texttt{=>}\ d$ for the recursive function,

$$\texttt{rec}\ f\ x\ \texttt{=>}\ d \stackrel{\text{def}}{=} \texttt{fn}\ x\ \texttt{=>}\ \texttt{let fun}\ f\ x = d\ \texttt{in}\ d\ \texttt{end}.$$

For example, $\texttt{fn}\ x\ \texttt{=>}\ \Omega = \texttt{rec}\ f\ x\ \texttt{=>}\ f\ f$.

A call-by-value fixed point combinator, $\texttt{Yv}$, can be expressed as:

$$\texttt{Yv} \stackrel{\text{def}}{=} \texttt{fn}\ f\ \texttt{=>}\ (\texttt{fn}\ g\ \texttt{=>}\ \texttt{fn}\ x\ \texttt{=>}\ f\ (g\ g)\ x)\ (\texttt{fn}\ g\ \texttt{=>}\ \texttt{fn}\ x\ \texttt{=>}\ f\ (g\ g)\ x).$$

So explicit recursion is redundant; later on we prove $\texttt{rec}\ f\ x\ \texttt{=>}\ e$ is semantically equivalent to $\texttt{Yv}\,(\texttt{fn}\ f\ \texttt{=>}\ \texttt{fn}\ x\ \texttt{=>}\ e)$.

Evaluation semantics

We define the operational semantics of closed expressions by an evaluation relation $\Downarrow\ \subseteq\ \mathit{Exp}_\emptyset \times \mathit{Val}_\emptyset$ between expressions and values. An important measure of ‘computational cost’ which we shall use extensively is the number of function applications (‘computation steps’) in evaluations. Therefore we introduce a family of evaluation relations indexed by this measure, $\Downarrow_N\ \subseteq\ \mathit{Exp}_\emptyset \times \mathit{Val}_\emptyset$ for $N \geq 0$, inductively defined by the rules in Table 1.

‘Plain’ evaluation is just $\Downarrow \stackrel{\text{def}}{=} \bigcup_{N\geq 0} \Downarrow_N$, i.e., $p \Downarrow v$ iff $\exists N \geq 0.\ p \Downarrow_N v$. It is also given inductively by Table 1 with all N subscripts erased from the rules.

Note that $v \Downarrow_0 v$ for all $v \in \mathit{Val}_\emptyset$. Moreover, evaluation is deterministic:

Proposition 1 (Determinacy) *If $p \Downarrow_N v$ and $p \Downarrow_{N'} v'$, $N = N'$ and $v = v'$.*

Examples

1. Let $\texttt{I}$ be the identity function, $\texttt{I} \stackrel{\text{def}}{=} \texttt{fn}\ x\ \texttt{=>}\ x$, then

$$\texttt{I}\ p \Downarrow_{N+1} v \quad \text{iff} \quad p \Downarrow_N v,$$

for all $p \in \mathit{Exp}_\emptyset$, $v \in \mathit{Val}_\emptyset$ and $N \geq 0$. (Therefore we shall use applications of $\texttt{I}$ as syntactic representations of function application steps in analogy with Sands’ ‘ticks’ (1997b).)

$$\text{(Eval fn)}\ \ \texttt{fn}\ x\ \texttt{=>}\ e\ \Downarrow_0\ \texttt{fn}\ x\ \texttt{=>}\ e$$

$$\text{(Eval nil)}\ \ \texttt{nil}\ \Downarrow_0\ \texttt{nil}$$

$$\text{(Eval cons)}\ \frac{e_1 \Downarrow_{N_1} v_1 \quad e_2 \Downarrow_{N_2} v_2}{e_1\ \texttt{::}\ e_2\ \Downarrow_{N_1+N_2}\ v_1\ \texttt{::}\ v_2}$$

$$\text{(Eval apply)}\ \frac{e_1 \Downarrow_{N_1} \texttt{fn}\ x\ \texttt{=>}\ e \quad e_2 \Downarrow_{N_2} v_2 \quad e\{v_2/x\} \Downarrow_{N_3} v}{e_1\ e_2\ \Downarrow_{N_1+N_2+N_3+1}\ v}$$

$$\text{(Eval let fun)}\ \frac{e\{(\texttt{rec}\ f\ x\ \texttt{=>}\ d)/f\} \Downarrow_N v}{\texttt{let fun}\ f\ x = d\ \texttt{in}\ e\ \texttt{end}\ \Downarrow_N\ v}$$

$$\text{(Eval let val)}\ \frac{d \Downarrow_{N_1} u \quad e\{u/x\} \Downarrow_{N_2} v}{\texttt{let val}\ x = d\ \texttt{in}\ e\ \texttt{end}\ \Downarrow_{N_1+N_2}\ v}$$

$$\text{(Eval case)}\ \frac{e_0 \Downarrow_{N_1} v_0 \quad e \Downarrow_{N_2} v \quad e = \begin{cases} e_1 & \text{if } v_0 = \texttt{nil} \\ e_2\{v_1/x_1, v_2/x_2\} & \text{if } v_0 = v_1\ \texttt{::}\ v_2 \\ e_3\{v_0/f\} & \text{if } v_0 = \texttt{fn}\ x\ \texttt{=>}\ d \end{cases}}{\begin{pmatrix} \texttt{case}\ e_0\ \texttt{of nil =>}\ e_1 \\ \quad\quad\ \mid\ x_1\ \texttt{::}\ x_2\ \texttt{=>}\ e_2 \\ \quad\quad\ \mid\ f\ \texttt{=>}\ e_3 \end{pmatrix} \Downarrow_{N_1+N_2}\ v}$$

Table 1: Evaluation relation

2. The divergent expression Ω does not evaluate to anything. Any derivation $\Omega \Downarrow_N v$ would have $\Omega \Downarrow_{N-1} v$ as premise, and this is impossible because of determinacy.

3. Let $e^\infty \stackrel{\text{def}}{=} (\texttt{fn}\ g\ \texttt{=>}\ \texttt{fn}\ x\ \texttt{=>}\ e(g\,g)x)\ (\texttt{fn}\ g\ \texttt{=>}\ \texttt{fn}\ x\ \texttt{=>}\ e(g\,g)x)$, so that $\texttt{Yv} = \texttt{fn}\ f\ \texttt{=>}\ f^\infty$. Then

$$\texttt{Yv}\ u \Downarrow_2 \texttt{fn}\ x\ \texttt{=>}\ u\ u^\infty\ x \quad \text{because} \quad u^\infty \Downarrow_1 \texttt{fn}\ x\ \texttt{=>}\ u\ u^\infty\ x,$$

for $u \in \mathit{Val}_\emptyset$.

Let an *evaluation context*, E, be a term with a hole, $-$, at redex position (Felleisen and Friedman 1987). They are given by the grammar:

$$\begin{array}{llll} (\textit{Ev. ctx.}) & E & ::= & - \mid E\ \texttt{::}\ e \mid e\ \texttt{::}\ E \mid E\ e \mid e\ E \\ & & \mid & \texttt{let val}\ x = E\ \texttt{in}\ e\ \texttt{end} \\ & & \mid & (\texttt{case}\ E\ \texttt{of nil =>}\ e_1 \mid x_1\ \texttt{::}\ x_2\ \texttt{=>}\ e_2 \mid f\ \texttt{=>}\ e_3). \end{array}$$

We write $E[e]$ for the term obtained from E by filling in e for the hole $-$. (We adopt a liberal definition of redex position which does not suggest an evaluation order in 'cons' expressions, $e_1\ \texttt{::}\ e_2$, and function applications, $e_1\ e_2$. One can indicate a left-to-right evaluation order by excluding evaluation contexts of the form $e\ \texttt{::}\ E$ and $e\ E$ where e is not a value. But evaluation order is immaterial here as we do not consider small-step reductions of terms and our language has no side effects.)

Evaluation contexts satisfy

$$E[p] \Downarrow_N v \quad \text{iff} \quad \exists M, u.\ p \Downarrow_M u\ \ \&\ \ E[u] \Downarrow_{N-M} v. \tag{2.1}$$

Combined with the examples above, we see that $E[\Omega]$ diverges and

$$E[\texttt{I}\ p] \Downarrow_{N+1} v \quad \text{iff} \quad \texttt{I}\ E[p] \Downarrow_{N+1} v \quad \text{iff} \quad E[p] \Downarrow_N v. \tag{2.2}$$

3 Relations

This section introduces our notation for relations and operations on them. Compatible refinement and context closure are of particular importance. Their precise definitions are key to the relational proofs in later sections. The relational algebra given here is quite general and language independent, except that only value substitutions are considered as our language is call-by-value.

Open and closed relations

A binary relation R is a set of pairs. We use infix notation, $a\ R\ b$, to mean $(a, b) \in R$.

Let Rel be the universal relation on closed expressions,

$$Rel = \{(p, p') \mid p, p' \in Exp_\emptyset\}.$$

We call every $R \subseteq Rel$ a *closed* relation. For instance, $Id = \{(p, p) \mid p \in Exp_\emptyset\}$ is the closed identity relation.

Moreover, we define

$$Rel^\circ = \{((\vec{x})e, (\vec{x})e') \mid e, e' \in Exp_{\vec{x}}\},$$

where $(\vec{x})e$ is a 'meta-abstraction' of $e \in Exp_{\vec{x}}$; the $(\vec{x})$ prefix is a binder and $\vec{x}$ is subject to α-renaming. We call all $R \subseteq Rel^\circ$ *open* relations and write $\vec{x} \vdash e \; R \; e'$ whenever $(\vec{x})e \; R \; (\vec{x})e'$. By identifying every $p \in Exp_\emptyset$ with the 0-ary abstraction $()p$, we have $Rel \subseteq Rel^\circ$ and closed relations are special cases of open ones.

We call Rel° the *open extension* of Rel. Generally, given any closed relation R, its open extension, $R^\circ \subseteq Rel^\circ$, is given by

$$\frac{\forall v_1, \ldots, v_n \in Val_\emptyset.\; e\{v_1 \cdots v_n/x_1 \ldots x_n\} \; R \; e'\{v_1 \cdots v_n/x_1 \ldots x_n\}}{x_1 \ldots x_n \vdash e \; R^\circ \; e'}$$

For example, Id° is the open identity relation.

Both Rel and Rel° are closed under relation composition, which we write by juxtaposition, $a \; R \, S \; b \overset{\text{def}}{\Leftrightarrow} \exists c.\; a \; R \; c \;\wedge\; c \; S \; b$. Open extension satisfies

$$R^\circ \, S^\circ \subseteq (R \, S)^\circ. \tag{3.1}$$

Relation Substitution

For $R, S \subseteq Rel^\circ$, the *relation substitution* of S into R, written $R\{S\} \subseteq Rel^\circ$, relates expressions obtained by simultaneous substitution of S related values into R related expressions,

$$\frac{\vec{x} \vdash e \; R \; e' \quad \vec{y} \vdash \vec{v} \; S \; \vec{v}'}{\vec{y} \vdash e\{\vec{v}/\vec{x}\} \; R\{S\} \; e'\{\vec{v}'/\vec{x}\}}$$

where $\vec{y} \vdash \vec{v} \; S \; \vec{v}'$ is shorthand for $\vec{y} \vdash v_i \; S \; v'_i$, for all $i = 1 \ldots n$, if $\vec{v} = v_1 \ldots v_n$ and $\vec{v}' = v'_1 \ldots v'_n$. Relation substitution is associative. Note that $R\{S\} \subseteq Rel$ if $S \subseteq Rel$. As a drill in the notation let us show

$$R \subseteq S^\circ \quad \text{iff} \quad R\{Id\} \subseteq S. \tag{3.2}$$

For the forward implication, suppose $R \subseteq S^\circ$ and $e\{\vec{v}/\vec{x}\} \; R\{Id\} \; e'\{\vec{v}/\vec{x}\}$ because $\vec{x} \vdash e \; R \; e'$ and $v_1, \ldots, v_n \in Val_\emptyset$. Then $\vec{x} \vdash e \; S^\circ \; e'$ and, by definition of open extension, $e\{\vec{v}/\vec{x}\} \; S \; e'\{\vec{v}/\vec{x}\}$. Conversely, if $R\{Id\} \subseteq S$ and $\vec{x} \vdash e \; R \; e'$ then $e\{\vec{v}/\vec{x}\} \; R\{Id\} \; e'\{\vec{v}/\vec{x}\}$ and $e\{\vec{v}/\vec{x}\} \; S \; e'\{\vec{v}/\vec{x}\}$, for all $v_1, \ldots, v_n \in Val_\emptyset$. From the definition of open extension we get $\vec{x} \vdash e \; S^\circ \; e'$, as required.

$$\text{(Comp } x\text{)}\ \vec{x}x\vec{y} \vdash x\ \widehat{R}\ x$$

$$\text{(Comp fn)}\ \frac{\vec{x}x \vdash e\ R\ e'}{\vec{x} \vdash \texttt{fn}\ x\ \texttt{=>}\ e\ \ \widehat{R}\ \ \texttt{fn}\ x\ \texttt{=>}\ e'}$$

$$\text{(Comp nil)}\ \vec{x} \vdash \texttt{nil}\ \widehat{R}\ \texttt{nil}$$

$$\text{(Comp cons)}\ \frac{\vec{x} \vdash e_1\ R\ e'_1 \quad \vec{x} \vdash e_2\ R\ e'_2}{\vec{x} \vdash e_1\ \texttt{::}\ e_2\ \ \widehat{R}\ \ e'_1\ \texttt{::}\ e'_2}$$

$$\text{(Comp apply)}\ \frac{\vec{x} \vdash e_1\ R\ e'_1 \quad \vec{x} \vdash e_2\ R\ e'_2}{\vec{x} \vdash e_1\ e_2\ \ \widehat{R}\ \ e'_1\ e'_2}$$

$$\text{(Comp let fun)}\ \frac{\vec{x}fx \vdash d\ R\ d' \quad \vec{x}f \vdash e\ R\ e'}{\vec{x} \vdash \texttt{let fun}\ f\ x = d\ \texttt{in}\ e\ \texttt{end}\ \ \widehat{R}\ \ \texttt{let fun}\ f\ x = d'\ \texttt{in}\ e'\ \texttt{end}}$$

$$\text{(Comp let val)}\ \frac{\vec{x} \vdash d\ R\ d' \quad \vec{x}x \vdash e\ R\ e'}{\vec{x} \vdash \texttt{let val}\ x = d\ \texttt{in}\ e\ \texttt{end}\ \ \widehat{R}\ \ \texttt{let val}\ x = d'\ \texttt{in}\ e'\ \texttt{end}}$$

$$\text{(Comp case)}\ \frac{\vec{x} \vdash d\ R\ d' \quad \vec{x} \vdash e_1\ R\ e'_1 \quad \vec{x}x_1x_2 \vdash e_2\ R\ e'_2 \quad \vec{x}f \vdash e_3\ R\ e'_3}{\vec{x} \vdash \begin{pmatrix} \texttt{case}\ d\ \texttt{of nil =>}\ e_1 \\ \quad\quad \mid x_1\ \texttt{::}\ x_2\ \texttt{=>}\ e_2 \\ \quad\quad \mid f\ \texttt{=>}\ e_3 \end{pmatrix}\ \widehat{R}\ \begin{pmatrix} \texttt{case}\ d'\ \texttt{of nil =>}\ e'_1 \\ \quad\quad \mid x_1\ \texttt{::}\ x_2\ \texttt{=>}\ e'_2 \\ \quad\quad \mid f\ \texttt{=>}\ e'_3 \end{pmatrix}}$$

Table 2: Compatible refinement

For any open relation R, we say R satisfies *weakening* if

$$\vec{x}\vec{y} \vdash e\ R\ e' \ \Rightarrow\ \vec{x}x\vec{y} \vdash e\ R\ e', \text{ if } x \notin \vec{x}\vec{y}.$$

We call R *substitutive* if $R\{R\} \subseteq R$, and we say that R is *closed under substitutions* if $R\{Id^\circ\} \subseteq R$. In the latter case $R\{Id\} = R \cap Rel$. Every open extension, R°, satisfies weakening and is closed under substitutions. Any substitutive and reflexive open relation also satisfies weakening and closure under substitutions. Each of these properties is preserved by relation composition.

Compatible refinement

For every open relation R, its *compatible refinement* (Gordon 1994) $\widehat{R} \subseteq Rel^\circ$ relates expressions with identical outermost syntactic constructor and immediate subterms pairwise related by R. Table 2 makes this definition precise for our language. Compatible refinement is monotone, preserves weakening, and commutes with relation composition, $\widehat{R\,S} = \widehat{R}\,\widehat{S}$.

An open relation R is *compatible* if $\widehat{R} \subseteq R$. Every compatible relation is reflexive, as can be shown by structural induction on expressions.

Compatibility can also be expressed in terms of contexts. A *context* C is an expression with 'holes'. If C has n holes, $C[e_1,\ldots,e_n]$ denotes the expression obtained by filling expressions $e_1 \ldots e_n$ into the holes in C, possibly involving capture of free variables of e_i if the i'th hole occurs in the scope of binders in C, for $i = 1,\ldots,n$. A relation R is compatible if whenever e_i and e'_i are related by R, for $i = 1,\ldots,n$, so are $C[e_1,\ldots,e_n]$ and $C[e'_1,\ldots,e'_n]$, for all contexts C. But a precise formulation of this which accounts for free variables and variable capture becomes complicated. The formalisation above, $\widehat{R} \subseteq R$, is easier to work with.

Throughout, we exploit compatible refinement as a tractable, indirect notation for contexts.

Lemma 1 *Any compatible and transitive relation which is closed under substitutions is substitutive.*

Proof Suppose R is compatible, transitive and closed under substitutions. If $\vec{x} \vdash e\ R\ e'$ and $\vec{y} \vdash \vec{u}\ R\ \vec{u}'$, then $\vec{y} \vdash e\{\vec{u}/\vec{x}\}\ Id^\circ\{R\}\ e\{\vec{u}'/\vec{x}\}$ and $\vec{y} \vdash e\{\vec{u}'/\vec{x}\}\ R\{Id^\circ\}\ e'\{\vec{u}'/\vec{x}\}$. Since R is compatible, $\vec{y} \vdash e\{\vec{u}/\vec{x}\}\ R\ e\{\vec{u}'/\vec{x}\}$ follows by easy structural induction on e; in general, $Id^\circ\{R\} \subseteq R$ for any compatible relation R. Moreover, $\vec{y} \vdash e\{\vec{u}'/\vec{x}\}\ R\ e'\{\vec{u}'/\vec{x}\}$ since R is closed under substitutions. By transitivity we conclude $\vec{y} \vdash e\{\vec{u}/\vec{x}\}\ R\ e'\{\vec{u}'/\vec{x}\}$, as required. $\square$

Context closure

For any relation R, its *context closure*, $R^{\mathsf{C}} \subseteq Rel^\circ$, relates expressions e, e' with matching outermost context C,

$$e = C[d_1,\ldots,d_n], \quad e' = C[d'_1,\ldots,d'_n],$$

and subterms d_i, d'_i related by R. This can be defined inductively by means of compatible refinement,

$$\text{(Ctx } R\text{)}\ \frac{\vec{y} \vdash e\ R\ e'}{\vec{x} \vdash e\ R^{\mathsf{C}}\ e'}\ \text{if } \vec{y} \subseteq \vec{x} \qquad \text{(Ctx Comp)}\ \frac{\vec{x} \vdash e\ \widehat{R^{\mathsf{C}}}\ e'}{\vec{x} \vdash e\ R^{\mathsf{C}}\ e'}$$

where $\vec{y} \subseteq \vec{x}$ means that all variables in $\vec{y}$ occur in $\vec{x}$, in any order. The side condition $\vec{y} \subseteq \vec{x}$ ensures that R^{C} satisfies weakening, even if R does not. Furthermore, context closure is monotone, idempotent $(R^{\mathsf{C}})^{\mathsf{C}} = R^{\mathsf{C}}$, and R^{C} is compatible, by (Ctx Comp).

Lemma 2 *If R is closed then R^{C} is substitutive.*

Proof (Sketch) Whenever $\vec{x} \vdash e\ R^{\mathsf{C}}\ e'$ and $\vec{y} \vdash \vec{v}\ R^{\mathsf{C}}\ \vec{v}'$, we can prove $\vec{y} \vdash e\{\vec{v}/\vec{x}\}\ R^{\mathsf{C}}\ e'\{\vec{v}'/\vec{x}\}$ by induction on the derivation of $\vec{x} \vdash e\ R^{\mathsf{C}}\ e'$.

Weakening is used as we enter the scope of binders. For example, if $\vec{x} \vdash e\ R^{\mathsf{C}}\ e'$ is derived by (Ctx Comp) and (Comp fn), then $e = \texttt{fn}\ z\ \texttt{=>}\ d$, $e' = \texttt{fn}\ z\ \texttt{=>}\ d'$, and $\vec{x}z \vdash d\ R^{\mathsf{C}}\ d'$. By weakening, $\vec{y}z \vdash \vec{v}\ R^{\mathsf{C}}\ \vec{v}'$ holds. Furthermore, $\vec{y}z \vdash z\ R^{\mathsf{C}}\ z$, by (Comp x) and (Ctx Comp). We calculate

$$\begin{array}{lll}
\vec{x}z \vdash d\ R^{\mathsf{C}}\ d' \ \ \&\ \ \vec{y}z \vdash \vec{v}z\ R^{\mathsf{C}}\ \vec{v}'z & & \\
\quad\Rightarrow\quad \vec{y}z \vdash d\{\vec{v}z/\vec{x}z\}\ R^{\mathsf{C}}\ d'\{\vec{v}'z/\vec{x}z\} & & \text{by induction hypothesis} \\
\quad\Rightarrow\quad \vec{y}z \vdash d\{\vec{v}/\vec{x}\}\ R^{\mathsf{C}}\ d'\{\vec{v}'/\vec{x}\} & & \\
\quad\Rightarrow\quad \vec{y} \vdash (\mathtt{fn}\ z\ \texttt{=>}\ d\{\vec{v}/\vec{x}\})\ \widehat{R^{\mathsf{C}}}\ (\mathtt{fn}\ z\ \texttt{=>}\ d'\{\vec{v}'/\vec{x}\}) & & \text{by (Comp fn)} \\
\quad\Rightarrow\quad \vec{y} \vdash e\{\vec{v}/\vec{x}\}\ R^{\mathsf{C}}\ e'\{\vec{v}'/\vec{x}\} & & \text{by (Ctx Comp).}
\end{array}$$

$\square$

Substitutive context closure, R^{SC}, is a substitutive extension of ordinary context closure, R^{C}. Each R^{SC} relates expressions e, e' with matching outermost context C,

$$e = C[d_1\{\vec{v_1}/\vec{x}\}, \ldots, d_n\{\vec{v_n}/\vec{x}\}], \quad e' = C[d_1'\{\vec{v_1}'/\vec{x}\}, \ldots, d_n'\{\vec{v_n}'/\vec{x}\}],$$

subterms d_i', d_i' related by R, and substitutions with values $\vec{v_i}, \vec{v_i}'$ inductively related by R^{SC}. It is important that R^{SC} has a succinct inductive definition,

$$\text{(SC Subst)}\ \frac{\vec{x} \vdash e\ R\{R^{\mathsf{SC}}\}\ e'}{\vec{x} \vdash e\ R^{\mathsf{SC}}\ e'} \qquad \text{(SC Comp)}\ \frac{\vec{x} \vdash e\ \widehat{R^{\mathsf{SC}}}\ e'}{\vec{x} \vdash e\ R^{\mathsf{SC}}\ e'}$$

Clearly $R^{\mathsf{C}} \subseteq R^{\mathsf{SC}}$ and if R is closed they coincide. The advantage of R^{SC} is that it is always substitutive. Substitutive context closure is monotone, idempotent, and R^{SC} is compatible, substitutive, and satisfies weakening. Compatibility is direct from (SC Comp). Weakening and substitutivity follow by induction on derivations. Since R^{SC} is compatible and substitutive, it is also reflexive and closed under substitutions.

Readers familiar with 'meta-terms' (Klop, van Oostrom, and van Raamsdonk 1993) will notice that substitutive context closure corresponds to closure under substitution of related meta-abstractions for meta-variables in meta-terms, whereas ordinary context closure is the closure under conventional variable capturing contexts. In fact, Pitts (1994b) advocates meta-terms, called 'extended expressions', as a generalised notion of contexts in place of conventional variable capturing contexts because the latter cannot be identified up to α-renaming of bound variables. However, our relational representation of contexts allows us to reason about conventional variable capturing contexts up to α-equivalence.

4 The unwinding theorem

As a first illustration of our relational approach to reasoning about contexts, we give a relational proof of the unwinding theorem. It says that a recursive function in a context converges if and only if one of its finite approximants does. The *finite approximants* of $\mathtt{rec}\ f\ x\ \texttt{=>}\ d$ are given inductively by

$$\begin{array}{lcl}
\mathtt{rec}^{(0)}\ f\ x\ \texttt{=>}\ d & \stackrel{\text{def}}{=} & \mathtt{fn}\ x\ \texttt{=>}\ \Omega, \\
\mathtt{rec}^{(n+1)}\ f\ x\ \texttt{=>}\ d & \stackrel{\text{def}}{=} & \mathtt{fn}\ x\ \texttt{=>}\ \mathtt{let}\ \mathtt{val}\ f = (\mathtt{rec}^{(n)}\ f\ x\ \texttt{=>}\ d)\ \mathtt{in}\ d\ \mathtt{end}.
\end{array}$$

We say p *converges* iff $\exists v.\, p \Downarrow v$.

Theorem 1 (Unwinding) *For every recursive function* $\texttt{rec}\ f\ x\ \texttt{=>}\ d$ *and every context* C, $C[\texttt{rec}\ f\ x\ \texttt{=>}\ d]$ *converges if and only if there exists* $n \geq 0$ *such that* $C[\texttt{rec}^{(n)}\ f\ x\ \texttt{=>}\ d]$ *converges.*

Our proof below shows how the relational notation offers a tractable formulation of a complex syntactic argument. For instance, the proof is not complicated by the fact that we prove the theorem for arbitrary recursive functions, possibly with free variables.

First we construct a family of relations $\{R_n\}_{n\geq 0}$ with each R_n given by

$$\vec{x} \vdash \texttt{rec}\ f\ x\ \texttt{=>}\ d\ \ R_n\ \ \texttt{rec}^{(n)}\ f\ x\ \texttt{=>}\ d,$$

$$\vec{x} \vdash \texttt{let fun}\ f\,x = d\ \texttt{in}\ d\ \texttt{end}\ \ R_n\ \ \texttt{let val}\ f = (\texttt{rec}^{(n)}\ f\ x\ \texttt{=>}\ d)\ \texttt{in}\ d\ \texttt{end},$$

if $d \in \mathit{Exp}_{\vec{x}fx}$. For each $n \geq 0$, we construct a relation U_n which satisfies

$$\vdash C[\texttt{rec}\ f\ x\ \texttt{=>}\ d]\ \ U_n\ \ C[\texttt{rec}^{(n)}\ f\ x\ \texttt{=>}\ d], \tag{4.1}$$

for arbitrary contexts C. In the course of the proof of the main lemma below, U_n must be preserved by evaluation in an appropriate sense. Therefore we cannot take U_n to be the context closure of R_n. We are going to strengthen the induction hypothesis by taking U_n to be the larger relation

$$U_n \stackrel{\text{def}}{=} \left(\bigcup_{m\geq n} R_m\right)^{\mathsf{SC}}.$$

By this definition, U_n satisfies (4.1), it is substitutive, and $U_n \subseteq U_{n'}$ whenever $n' \leq n$. These are key properties for the proof that are easier to formulate precisely in terms of relations rather than contexts. The inductive definition of substitutive context closure is also convenient for formal reasoning. By the construction of U_n, whenever $\vec{x} \vdash e\ U_n\ e'$, we can argue by cases on the derivation: either $\vec{x} \vdash e\ R_m\{U_n\}\ e'$ for some $m \geq n$, by (SC Subst), and we can decompose e and e' into expressions related by R_m and substitutions of values related by U_n; or $\vec{x} \vdash e\ \widehat{U_n}\ e'$, by (SC Comp), and we may proceed by analysis of the derivation by the rules for compatible refinement in Table 2. For instance, we can deduce, for all values v and v',

$$\vec{x} \vdash v\ U_{n+1}\ v' \quad \text{implies} \quad \vec{x} \vdash v\ \widehat{U_n}\ v', \tag{4.2}$$

since $U_{n+1} \subseteq U_n$ and $\vec{x} \vdash \texttt{rec}\ f\ x\ \texttt{=>}\ d\ \ \widehat{R_m}\ \ \texttt{rec}^{(m+1)}\ f\ x\ \texttt{=>}\ d$, for all m.

Lemma 3

(1) If $\vdash p\ U_{n+N}\ p'$ *and* $p \Downarrow_N v$, *also* $p' \Downarrow_N v'$ *and* $\vdash v\ U_n\ v'$, *for some* v'.

(2) If $\vdash p\ U_0\ p'$ *and* $p' \Downarrow_N v'$*, also* $p \Downarrow_N v$ *and* $\vdash v\ U_0\ v'$*, for some* v*.*

Proof In outline, the proof argument for (1) is that any occurrence of `rec` in p is "unfolded" (evaluated recursively) at most N times in the evaluation $p \Downarrow_N v$ and evaluates in "lock-step" with any $\texttt{rec}^{(m+N)}$ in p' ($m \geq n$). In the end, each residual occurrence of `rec` in v is matched by some $\texttt{rec}^{(m')}$ in v' ($m' \geq n$). The proof of (2) is similar; evaluation of $\texttt{rec}^{(m)}$ in p' is matched by evaluation of `rec` in p such that any residual occurrence $\texttt{rec}^{(m')}$ in v' is matched by `rec` in v.

We spell out the proof of (1) in detail as illustration of the relational proof technique explored in this paper. By induction on the derivation of $p \Downarrow_N v$, we inductively construct a related derivation $p' \Downarrow_N v'$. Consider the derivation of $\vdash p\ U_{n+N}\ p'$. There are two cases:

(SC Subst) $\vdash p\ R_m\{U_{n+N}\}\ p'$, for some $m \geq n + N$, and $p = r\{\vec{u}/\vec{x}\}$, $p' = r_m\{\vec{u}'/\vec{x}\}$, where $\vec{x} \vdash r\ R_m\ r_m$ and $\vdash \vec{u}\ U_{n+N}\ \vec{u}'$.

If $r = \texttt{rec}\ f\ x \texttt{ => } d$, $r_m = \texttt{rec}^{(m)}\ f\ x \texttt{ => } d$, then p, p' are values, $N = 0$, $v = p$, $p' \Downarrow_0 v' = p'$, and $\vdash v = p\ U_n\ p' = v'$.

If $r = \texttt{let fun}\ f\ x = d \texttt{ in } e \texttt{ end}$, $r_m = \texttt{let fun}^{(m)}\ f\ x = d \texttt{ in } e \texttt{ end}$, then $p \Downarrow_N v$ must be derived by rule (Eval let fun) from $e\{\vec{u}t/\vec{x}f\} \Downarrow_N v$, where $t = (\texttt{rec}\ f\ x \texttt{ => } d)\{\vec{u}/\vec{x}\}$. We let $t' = (\texttt{rec}^{(m)}\ f\ x \texttt{ => } d)\{\vec{u}'/\vec{x}\}$ and observe that $\vdash t\ U_{n+N}\ t'$ and $\vdash e\{\vec{u}t/\vec{x}f\}\ U_{n+N}\ e\{\vec{u}'t'/\vec{x}f\}$. By induction hypothesis $e\{\vec{u}'t'/\vec{x}f\} \Downarrow_N v'$ with $\vdash v\ U_n\ v'$. And from the definition of r_m and by (Eval let val), also $p' \Downarrow_N v'$.

(SC Comp) $\vdash p\ \widehat{U_{n+N}}\ p'$. We proceed by analysis of the derivation of $p \Downarrow_N v$.

Case (Eval apply) $p = p_1\, p_2$, $p_1 \Downarrow_{N_1} v_1 = \texttt{fn}\ x \texttt{ => } e$, $p_2 \Downarrow_{N_2} v_2$, $e\{v_2/x\} \Downarrow_{N_3} v$, and $N = N_1 + N_2 + N_3 + 1$. By (Comp apply), $p' = p_1'\, p_2'$ with $\vdash p_i\ U_{n+N}\ p_i'$ and by the induction hypothesis $p_i' \Downarrow_{N_i} v_i'$ with $\vdash v_i\ U_{n+N-N_i}\ v_i'$. Notice that $n + N - N_i > n + N_3 + 1$, for $i = 1, 2$. Therefore $\vdash v_2\ U_{n+N_3}\ v_2'$ and from (4.2) follows $v_1' = \texttt{fn}\ x \texttt{ => } e'$ such that $x \vdash e\ U_{n+N_3}\ e'$. So $\vdash e\{v_2/x\}\ U_{n+N_3}\ e'\{v_2'/x\}$, by substitutivity, and $e'\{v_2'/x\} \Downarrow_{N_3} v'$ with $\vdash v\ U_n\ v'$, by induction hypothesis. By (Eval apply), we conclude $p' \Downarrow_N v'$.

Case (Eval let fun) $p = \texttt{let fun}\ f\ x = e_1 \texttt{ in } e_2 \texttt{ end}$ and $e_2\{(\texttt{rec}\ f\ x \texttt{ => } e_1)/f\} \Downarrow_N v$. We have $p' = \texttt{let fun}\ f\ x = e_1' \texttt{ in } e_2' \texttt{ end}$, with $f x \vdash e_1\ U_{n+N}\ e_1'$ and $f \vdash e_2\ U_{n+N}\ e_2'$. Then also

$$\vdash (\texttt{rec}\ f\ x \texttt{ => } e_1)\ U_{n+N}\ (\texttt{rec}\ f\ x \texttt{ => } e_1')$$

and

$$\vdash e_2\{(\texttt{rec}\ f\ x \texttt{ => } e_1)/f\}\ U_{n+N}\ e_2'\{(\texttt{rec}\ f\ x \texttt{ => } e_1')/f\}$$

by compatibility and substitutivity of U_{n+N}. By the induction hypothesis $e_2'\{(\texttt{rec}\ f\ x \texttt{ => } e_1')/f\} \Downarrow_N v'$ such that $\vdash v\ U_n\ v'$. By (Eval let fun), we conclude $p' \Downarrow_N v'$.

Case (Eval case) $p = \texttt{case}\, p_0\, \texttt{of nil =>}\ e_1 \mid x_1 \texttt{ :: } x_2 \texttt{ =>}\ e_2 \mid f \texttt{ =>}\ e_3$, $p_0 \Downarrow_{N_1} u$, $q \Downarrow_{N_2} v$, $N = N_1 + N_2$, where

$$q = \begin{cases} e_1 & \text{if } u = \texttt{nil} \\ e_2\{u_1/x_1, u_2/x_2\} & \text{if } u = u_1 \texttt{ :: } u_2 \\ e_3\{u/f\} & \text{if } u = \texttt{fn}\ x \texttt{ =>}\ e. \end{cases}$$

By (Comp case), $p' = \texttt{case}\, p'_0\, \texttt{of nil =>}\ e'_1 \mid x_1 \texttt{ :: } x_2 \texttt{ =>}\ e'_2 \mid f \texttt{ =>}\ e'_3$, with $\vdash p_0\ U_{n+N}\ p'_0$, $\vdash e_1\ U_{n+N}\ e'_1$, $x_1 x_2 \vdash e_2\ U_{n+N}\ e'_2$, $f \vdash e_3\ U_{n+N}\ e'_3$. By induction hypothesis $p'_0 \Downarrow_{N_1} u'$ such that $\vdash u\ U_{n+N_2}\ u'$, since $n + N = (n+N_2)+N_1$. By analysis of the derivation of $\vdash u\ U_{n+N_2}\ u'$ we see that they have matching outermost constructor and, if this is not function abstraction, also $\vdash u\ \widehat{U_{n+N_2}}\ u'$. Accordingly, let

$$q' = \begin{cases} e'_1 & \text{if } u = \texttt{nil} = u' \\ e'_2\{u'_1/x_1, u'_2/x_2\} & \text{if } u = u_1 \texttt{ :: } u_2, u' = u'_1 \texttt{ :: } u'_2 \text{ with } \vdash u_i\ U_{n+N_2}\ u'_i \\ e'_3\{u'/f\} & \text{if } u = \texttt{fn}\ x \texttt{ =>}\ e, u' = \texttt{fn}\ x \texttt{ =>}\ e'. \end{cases}$$

Since $N_2 \le N$, $U_{n+N} \subseteq U_{n+N_2}$ and, by substitutivity of U_{n+N_2}, $\vdash q\ U_{n+N_2}\ q'$. By induction hypothesis $q' \Downarrow_{N_2} v'$ with $\vdash v\ U_n\ v'$ and we conclude $p' \Downarrow_N v'$ by (Eval case).

The remaining cases are simpler. This completes the proof of (1).

The proof of (2) is very similar and proceeds by induction on the derivation of $p' \Downarrow_N v'$. The (Eval apply) case exploits that the applied function cannot be $\texttt{rec}^{(0)}\ f\ x \texttt{ =>}\ e$ as this would diverge. □

Proof of The Unwinding Theorem If $C[\texttt{rec}\ f\ x \texttt{ =>}\ d] \Downarrow_N v$, from (4.1) and Lemma 3(1) follows, for all $n \ge N$, $\exists v'.\ C[\texttt{rec}^{(n)}\ f\ x \texttt{ =>}\ d] \Downarrow_N v'$. Conversely, if $C[\texttt{rec}^{(n)}\ f\ x \texttt{ =>}\ d] \Downarrow_N v'$, also $\exists v.\ C[\texttt{rec}\ f\ x \texttt{ =>}\ d] \Downarrow_N v$, by (4.1) and Lemma 3(2) because $U_n \subseteq U_0$. □

The backward direction of the proof can also be derived from the (computationally adequate) theory of applicative bisimulation below, instead of Lemma 3(2).

An important consequence of the unwinding theorem is a 'syntactic continuity' property of contextual equivalence (Pitts 1997). In Sections 5.4 and 6.3 we see how Lemma 3 entails syntactic continuity for applicative similarity and improvement. Syntactic continuity is a 'domain-theoretic' property that holds in all computationally adequate continuous models; see Pitts (1996a) and Braüner (1996).

There exist a number of operational proofs of these results, both for small-step reduction semantics (Mason, Smith, and Talcott 1996; Sands 1997a) and big-step evaluation semantics like ours (Pitts 1997). But note that our proof holds for *open* recursive terms r and that Lemma 3 gives very precise information about the operational relationship between r and its finite approximants. Our relational notation makes it feasible to express and reason about the details of contexts and substitutions. A characteristic of such relational proofs is that operational issues are dealt with in one sweeping induction on the derivation of evaluations and syntactic issues are dealt with in terms of the general algebra of relations. No auxiliary lemmas about evaluation and contexts are needed.

In the remainder of the paper we apply the relational technique used in the proof above to the study of operational preorders and equivalences.

5 Similarity

The primary operational relation we study is Abramsky's applicative bisimulation (Abramsky 1990). It is the basis for a co-inductive generalisation of Milner's context lemma (Milner 1977) to untyped functional languages. The basic idea is that higher-order functions are infinite data structures, built from the 'lazy' function abstraction data constructor, and are related co-inductively by applicative bisimulation in analogy with bisimulation of infinite behaviours in process calculi.

In this section we develop the theory of applicative (bi)simulation for our language, including preliminary simulation up to context results based on Howe's and Pitts' congruence proof techniques. This part is mainly a presentation of unpublished work by Pitts (1995) and serves as a basis for our further developments of this idea in Sections 6 and 7. Our aim is to develop techniques for reasoning about recursion. We shall see that simulation up to context is particularly useful for this purpose. In order to complete the discussion of proof rules for recursion we also prove a syntactic continuity property.

We consider an applicative bisimulation preorder, $\lesssim \subseteq Rel$, which we call *similarity*. Expressions are similar if they evaluate to similar values.

$$p \lesssim p' \quad \text{iff} \quad \forall v.\, p \Downarrow v \Rightarrow \exists v'.\, p' \Downarrow v' \ \ \&\ \ v \lesssim v'. \tag{5.1}$$

Functions are similar if they are similar on all arguments; by definition of open extension this may be expressed as

$$\texttt{fn}\, x \texttt{ => } e \quad \lesssim \quad \texttt{fn}\, x \texttt{ => } e' \quad \text{iff} \quad x \vdash e \lesssim^{\circ} e'. \tag{5.2}$$

Following Howe (1996) we extend similarity to arbitrary values with matching outermost constructor and immediate subterms pairwise similar.

$$\texttt{nil} \quad \lesssim \quad \texttt{nil}. \tag{5.3}$$

$$v_1 \texttt{ :: } v_2 \quad \lesssim \quad v_1' \texttt{ :: } v_2' \quad \text{iff} \quad v_1 \lesssim v_1' \quad \text{and} \quad v_2 \lesssim v_2'. \tag{5.4}$$

We regard (5.2)–(5.4) as a definition of similarity on values by structural induction. We formalise this by means of a variant of compatible refinement on values, akin to Gordon's 'matching values' (Gordon 1995). For every open relation R, let $\overline{R}$ relate 'matching' values built from identical value constructors and with function

$$\text{(Match } x\text{)}\ \vec{x}x\vec{y} \vdash x\ \overline{R}\ x$$

$$\text{(Match fn)}\ \frac{\vec{x}x \vdash e\ R\ e'}{\vec{x} \vdash \texttt{fn}\ x\ \texttt{=>}\ e\ \ \overline{R}\ \ \texttt{fn}\ x\ \texttt{=>}\ e'}$$

$$\text{(Match nil)}\ \vec{x} \vdash \texttt{nil}\ \overline{R}\ \texttt{nil}$$

$$\text{(Match cons)}\ \frac{\vec{x} \vdash v_1\ \overline{R}\ v_1' \quad \vec{x} \vdash v_2\ \overline{R}\ v_2'}{\vec{x} \vdash v_1\ \texttt{::}\ v_2\ \ \overline{R}\ \ v_1'\ \texttt{::}\ v_2'}$$

Table 3: Matching values

bodies pairwise related by R. This is defined inductively by the rules in Table 3. Now (5.2)–(5.4) can be expressed by

$$v \lesssim v' \quad \text{iff} \quad \vdash v\ \overline{\lesssim^\circ}\ v'. \tag{5.5}$$

We take the mutually recursive equations (5.1) and (5.5) as a co-inductive definition of similarity. To make this definition precise, we define a monotone simulation operator on relations, $\langle \_ \rangle$, which maps any open relation $R \subseteq Rel^\circ$ to the closed relation $\langle R \rangle \subseteq Rel$ given by

$$p\ \langle R \rangle\ p' \quad \overset{\text{def}}{\Leftrightarrow} \quad \forall v.\, p \Downarrow v \quad \Rightarrow \quad \exists v'.\, p' \Downarrow v' \ \ \&\ \ \vdash v\ \overline{R}\ v'.$$

We define similarity co-inductively as the greatest fixed point of $\langle \_^\circ \rangle$,

$$\lesssim \quad \overset{\text{def}}{=} \quad \nu R. \langle R^\circ \rangle, \tag{5.6}$$

and *bisimilarity*, $\sim$, as the greatest symmetric fixed point, definable as

$$\sim \quad \overset{\text{def}}{=} \quad \nu R. \langle R^\circ \rangle \cap \langle (R^\circ)^{\text{op}} \rangle^{\text{op}},$$

where $a\ S^{\text{op}}\ b \ \overset{\text{def}}{\Leftrightarrow}\ b\ S\ a$, for every relation S. Closed relations form a complete lattice, ordered by subset inclusion, and $\langle \_^\circ \rangle$ and $\langle \_^\circ \rangle \cap \langle (\_^\circ)^{\text{op}} \rangle^{\text{op}}$ are monotone operations with respect to this ordering. The Tarski-Knaster fixed point theorem asserts that their greatest fixed points exist and are also greatest post-fixed points.

Since evaluation is deterministic, $\sim$ is the largest symmetric relation contained in $\lesssim$,

$$\sim \quad \overset{\text{def}}{=} \quad \lesssim \cap \lesssim^{\text{op}}.$$

Therefore it suffices to focus attention on the more primitive relation $\lesssim$. In particular, we shall only formulate simulation proof rules for $\lesssim$ and omit the obvious analogues for $\sim$.

Equations (5.1)–(5.4) hold because $\lesssim$ is a fixed point, $\lesssim\, = \langle \lesssim^\circ \rangle$. An immediate consequence of (5.1) is *computational adequacy* with respect to termination behaviour,

$$p \not\lesssim \Omega \quad \text{iff} \quad \exists v.\ p \Downarrow v. \tag{5.7}$$

We shall call $R \subseteq \mathit{Rel}$ a *simulation* if it is a post-fixed point of $\langle \_^\circ \rangle$, that is, $R \subseteq \langle R^\circ \rangle$. By the co-inductive definition (5.6), $\lesssim$ is the largest simulation, and we have the co-induction simulation rule:

$$\frac{R \subseteq \langle R^\circ \rangle}{R \subseteq\, \lesssim}$$

Simulation is a powerful proof technique. To prove two expressions similar, we exhibit a simulation containing them. For example, $\lesssim$ is reflexive because the identity relation is a simulation, $\mathit{Id} \subseteq \langle \mathit{Id}^\circ \rangle$, and $\lesssim$ is transitive because $\lesssim\,\lesssim$ is a simulation, $\lesssim\,\lesssim\, = \langle \lesssim^\circ \rangle \langle \lesssim^\circ \rangle \subseteq \langle \lesssim^\circ\, \lesssim^\circ \rangle \subseteq \langle (\lesssim\,\lesssim)^\circ \rangle$, where we use the fact that $\langle \_ \rangle$ satisfies

$$\langle R \rangle \langle S \rangle \subseteq \langle R\, S \rangle, \text{ for all } R, S \subseteq \mathit{Rel}^\circ. \tag{5.8}$$

Hence $\lesssim$ is a preorder and $\sim$ is an equivalence relation.

We call expressions p and p' *Kleene equivalent* if they both diverge or both evaluate to the same value (because of determinacy, each expression can evaluate to at most one value). Kleene equivalence is easily seen to be a symmetric simulation and hence is included in $\sim$. An immediate consequence is *soundness* of evaluation with respect to bisimilarity,

$$p \Downarrow v \quad \text{implies} \quad p \sim v. \tag{5.9}$$

Many useful program laws are instances of Kleene equivalence. For instance, beta laws such as

$$(\texttt{fn}\ x\ \texttt{=>}\ e)\ v \;\sim\; e\{^{v}\!/_{x}\}.$$

$$\texttt{let fun}\ f\ x = d\ \texttt{in}\ e\ \texttt{end} \;\sim\; e\{^{(\texttt{rec}\ f\ x\ \texttt{=>}\ d)}\!/_{f}\}.$$

These also hold for open expressions and $\sim^\circ$, by definition of open extension. From the beta law for recursive function declarations follows a fixed point law for recursive functions,

$$\texttt{rec}\ f\ x\ \texttt{=>}\ d \;\sim\; \texttt{fn}\ x\ \texttt{=>}\ d\{^{(\texttt{rec}\ f\ x\ \texttt{=>}\ d)}\!/_{f}\},$$

by (5.2) and the definition of $\texttt{rec}\ f\ x\ \texttt{=>}\ d$.

As an example of a co-inductive argument about recursive data, consider

$$u \stackrel{\text{def}}{=} \texttt{rec}\ f\ x\ \texttt{=>}\ f, \quad v \stackrel{\text{def}}{=} \texttt{rec}\ f\ x\ \texttt{=>}\ \texttt{fn}\ y\ \texttt{=>}\ f. \tag{5.10}$$

By definition of recursive functions, $u = \texttt{fn}\ x\ \texttt{=>}\ p$ and $v = \texttt{fn}\ x\ \texttt{=>}\ q$, where

$$\begin{aligned} p &\stackrel{\text{def}}{=} \texttt{let fun}\ f\ x = f\ \texttt{in}\ f\ \texttt{end}, \\ q &\stackrel{\text{def}}{=} \texttt{let fun}\ f\ x = (\texttt{fn}\ y\ \texttt{=>}\ f)\ \texttt{in}\ (\texttt{fn}\ y\ \texttt{=>}\ f)\ \texttt{end}. \end{aligned}$$

Observe that $p \Downarrow u$ and $q \Downarrow$ `fn` y `=>` v. Both u and v perpetually return a function no matter how many arguments they are applied to. To see that they are bisimilar, we construct the relation $R = \{(p,q),(p,v)\}$ satisfying $u\ \langle R^\circ \rangle\ v$ and $v\ \langle (R^\circ)^{\mathrm{op}} \rangle\ u$. Both R and R^{op} are simulations, hence $R \subseteq \sim$, and $u \sim v$ follows by definition of $\sim$ because $\langle \_ \rangle$ is monotone.

Similarity extends to open expressions by open extension, $\lesssim^\circ$. A manipulation of fixed points and monotone operators yields $\lesssim^\circ = \nu R.\langle R \rangle^\circ$. We call R an *open simulation* if $R \subseteq \langle R \rangle^\circ$, in which case $R \subseteq \lesssim^\circ$, by co-induction.

$$\frac{R \subseteq \langle R \rangle^\circ}{R \subseteq \lesssim^\circ} \tag{5.11}$$

By (3.2), $R \subseteq \langle R \rangle^\circ$ iff $R\{\mathit{Id}\} \subseteq \langle R \rangle$. If R is closed under substitutions then $R\{\mathit{Id}\} = R \cap \mathit{Rel}$ and R is an open simulation if $R \cap \mathit{Rel} \subseteq \langle R \rangle$.

5.1 Simulation up to context

Often when one wants to prove that a relation R is contained in similarity, $R \subseteq \lesssim$, either R is not itself a simulation or it is not possible to show this directly. The solution is to extend R to a larger relation S which is a simulation and thus $S \subseteq \lesssim$ and $R \subseteq \lesssim$. In fact, this is the co-inductive dual of "strengthening the induction hypothesis" in induction arguments. The proofs of syntactic continuity and precongruence are examples of this. In both cases the constructed relations are tailored to the respective problems. However, often the process of 'completing' R follows a common pattern. We shall investigate refined simulation rules which implicitly extend R so as to become a simulation. Gordon (1995) presents a number of such refinements of bisimulation for a typed, call-by-name functional language. One of these is Milner's *bisimulation up to bisimilarity* (Milner 1989). For $\lesssim$ this says:

Proposition 2 (Simulation up to $\lesssim$) $\dfrac{R \subseteq \langle \lesssim^\circ\ R^\circ\ \lesssim^\circ \rangle}{R \subseteq \lesssim}$

Proof If $R \subseteq \langle \lesssim^\circ\ R^\circ\ \lesssim^\circ \rangle$, then $\lesssim R \lesssim$ is a simulation:

$$\begin{array}{rcll}
\lesssim R \lesssim & \subseteq & \langle \lesssim^\circ \rangle\, \langle \lesssim^\circ\ R^\circ\ \lesssim^\circ \rangle\, \langle \lesssim^\circ \rangle & \text{as } \lesssim \text{ is a fixed point for } \langle _^\circ \rangle \\
& \subseteq & \langle \lesssim^\circ \lesssim^\circ R^\circ \lesssim^\circ \lesssim^\circ \rangle & \text{by (5.8)} \\
& \subseteq & \langle \lesssim^\circ\ R^\circ\ \lesssim^\circ \rangle & \lesssim^\circ \text{ is transitive} \\
& \subseteq & \langle (\lesssim R \lesssim)^\circ \rangle & \text{(3.1).}
\end{array}$$

Therefore $\lesssim R \lesssim\ \subseteq\ \lesssim$ and, since $\lesssim$ is reflexive, $R \subseteq \lesssim$. □

It is possible to derive an 'equational' simulation rule which does not involve the simulation operator, $\langle \_ \rangle$, nor the evaluation relation.

Proposition 3 $\dfrac{R \subseteq \lesssim \overline{R^\circ} \lesssim}{R \subseteq \lesssim}$

Proof The inclusion $\lesssim \overline{R^\circ} \lesssim \subseteq \langle \lesssim^\circ R^\circ \lesssim^\circ \rangle$ follows easily from the definitions of $\lesssim$ and $\langle \_ \rangle$, and the rule is an immediate consequence of Proposition 2. $\Box$

Recall the example (5.10), where

$$u = \texttt{rec}\, f\, x \texttt{ => } f, \quad v = \texttt{rec}\, f\, x \texttt{ => } \texttt{fn}\, y \texttt{ => } f.$$

An 'equational' proof of $u \sim v$ using Proposition 3 looks as follows. Let $R = \{(u, v), (u, \texttt{fn}\, y \texttt{ => } v)\}$. By the fixed point law for `rec` we calculate

$$\begin{array}{lllll} u & \sim & \texttt{fn}\, x \texttt{ => } u \;\; \overline{R^\circ} \;\; \texttt{fn}\, x \texttt{ => } \texttt{fn}\, y \texttt{ => } v & \sim & v, \\ u & \sim & \texttt{fn}\, x \texttt{ => } u \;\; \overline{R^\circ} \;\; \texttt{fn}\, y \texttt{ => } v, & & \end{array}$$

hence $R \subseteq \lesssim$ and $R^{\mathrm{op}} \subseteq \lesssim$, by symmetrical applications of Proposition 3. Therefore $R \subseteq \sim$ and $u \sim v$.

Proposition 3 is not a complete proof rule because it can only relate expressions p and p' where p' converges: if $p \lesssim \overline{R^\circ} \lesssim p'$, there exist values v and v' such that $p \lesssim v \; \overline{R^\circ} \; v' \lesssim p'$, and then p' converges, by (5.1). This prevents us from relating two diverging expressions. We can repair this deficiency as in Gordon (1994), by throwing in the singleton relation $\{(\Omega, \Omega)\}$,

$$\frac{R \subseteq \lesssim (\overline{R^\circ} \cup \{(\Omega, \Omega)\}) \lesssim}{R \subseteq \lesssim} \tag{5.12}$$

Another refinement of bisimulation known from process calculi is Sangiorgi's powerful *bisimulation up to context* (Sangiorgi 1994). Here and in ensuing sections we study variants of this proof principle for similarity and improvement.

As a first formulation of simulation up to context we have the following result for closed relations and context closure. The proof is adapted from Pitts (1995).

Proposition 4 (Simulation up to context) $\dfrac{R \subseteq \langle R^{\mathsf{C}} \rangle}{R \subseteq \lesssim}$

Proof Assume $R \subseteq \langle R^{\mathsf{C}} \rangle$. We shall prove

$$R^{\mathsf{C}} \cap Rel \subseteq \langle R^{\mathsf{C}} \rangle. \tag{5.13}$$

Since R is closed, R^{C} is substitutive, by Lemma 2. As R^{C} is also reflexive, it is closed under substitutions. Therefore (5.13) implies that R^{C} is an open simulation, $R^{\mathsf{C}} \subseteq \langle R^{\mathsf{C}} \rangle^\circ$, and then $R^{\mathsf{C}} \subseteq \lesssim^\circ$, by the open simulation rule (5.11). As $R \subseteq R^{\mathsf{C}}$, the result follows.

By definition of $\langle \_ \rangle$, (5.13) means that whenever $\vdash p \; R^{\mathsf{C}} \; p'$ and $p \Downarrow v$, there exists v' such that $p' \Downarrow v'$ and $\vdash v \; \overline{R^{\mathsf{C}}} \; v'$. The proof is by induction on the derivation of $p \Downarrow v$.

(Ctx R) If $\vdash p \ R \ p'$, then $p \ \langle R^{\mathsf{C}} \rangle \ p'$ is immediate from assumption $R \subseteq \langle R^{\mathsf{C}} \rangle$.

(Ctx Comp) Otherwise $\vdash p \ \widehat{R^{\mathsf{C}}} \ p'$ and we proceed by analysis of the derivation of $p \Downarrow v$. Each case is as in the proof of Lemma 3, except that we omit the arithmetic on N subscripts exercised there and we observe that the results are matching values. We show three representative cases:

Case (Eval cons) $p = p_1 \texttt{ :: } p_2$, $p_i \Downarrow v_i$, and $v = v_1 \texttt{ :: } v_2$. Since $\vdash p \ \widehat{R^{\mathsf{C}}} \ p'$, we have $\underline{p' = p'_1 \texttt{ :: } p'_2}$ with $\vdash p_i \ R^{\mathsf{C}} \ p'_i$. By induction hypothesis $\underline{p'_i} \Downarrow v'_i$ such that $\vdash v_i \ \overline{R^{\mathsf{C}}} \ v'_i$, for $i = 1, 2$, so $p' \Downarrow v'_1 \texttt{ :: } v'_2$ and $\vdash v_1 \texttt{ :: } v_2 \ \overline{R^{\mathsf{C}}} \ v'_1 \texttt{ :: } v'_2$, by (Eval cons) and (Match cons).

Case (Eval apply) $p = p_1 \, p_2$, $p_1 \Downarrow v_1 = \texttt{fn}\ x \texttt{ => } e$, $p_2 \Downarrow v_2$, and $e\{v_2/x\} \Downarrow v$. Since $\vdash p \ \widehat{R^{\mathsf{C}}} \ p'$, also $p' \ \underline{=\ p'_1 \, p'_2}$ with $\vdash p_i \ R^{\mathsf{C}} \ p'_i$. By the induction hypothesis $p'_i \Downarrow v'_i$ with $\vdash v_i \ \overline{R^{\mathsf{C}}} \ v'_i$, for $i = 1, 2$. Then $v'_1 = \texttt{fn}\ x \texttt{ => } e'$ with $x \vdash e \ R^{\mathsf{C}} \ e'$ and, by compatibility, $\vdash v_2 \ R^{\mathsf{C}} \ v'_2$. By substitutivity and induction hypothesis, $\vdash e\{v_2/x\} \ R^{\mathsf{C}} \ e'\{v'_2/x\}$ and $e'\{v'_2/x\} \Downarrow v'$ with $\vdash v \ \overline{R^{\mathsf{C}}} \ v'$. We conclude $p' \Downarrow v'$ by (Eval apply).

Case (Eval case) $p = \texttt{case}\ p_0 \texttt{ of nil => } e_1 \mid x_1 \texttt{ :: } x_2 \texttt{ => } e_2 \mid f \texttt{ => } e_3$, $p_0 \Downarrow u$ and $q \Downarrow v$, where

$$q = \begin{cases} e_1 & \text{if } u = \texttt{nil} \\ e_2\{u_1/x_1, u_2/x_2\} & \text{if } u = u_1 \texttt{ :: } u_2 \\ e_3\{u/f\} & \text{if } u = \texttt{fn}\ x \texttt{ => } d. \end{cases}$$

By (Comp case), $p' = \texttt{case}\ p'_0 \texttt{ of nil => } e'_1 \mid x_1 \texttt{ :: } x_2 \texttt{ => } e'_2 \mid f \texttt{ => } e'_3$, with $\vdash p_0 \ R^{\mathsf{C}} \ p'_0$, $\vdash e_1 \ R^{\mathsf{C}} \ e'_1$, $x_1 \underline{x_2} \vdash e_2 \ R^{\mathsf{C}} \ e'_2$, $f \vdash e_3 \ R^{\mathsf{C}} \ e'_3$. By induction hypothesis $p'_0 \Downarrow u'$ such that $\vdash u \ \overline{R^{\mathsf{C}}} \ u'$. By analysis of the derivation of the latter, we construct

$$q' = \begin{cases} e'_1 & \text{if } u = \texttt{nil} = u' \\ e'_2\{u'_1/x_1, u'_2/x_2\} & \text{if } u = u_1 \texttt{ :: } u_2, u' = u'_1 \texttt{ :: } u'_2 \text{ with } \vdash u_i \ \overline{R^{\mathsf{C}}} \ u'_i \\ e'_3\{u'/f\} & \text{if } u = \texttt{fn}\ x \texttt{ => } d, u' = \texttt{fn}\ x \texttt{ => } d', \end{cases}$$

where $\vdash q \ R^{\mathsf{C}}\{\overline{R^{\mathsf{C}}}\} \ q'$. By compatibility and substitutivity, $\overline{R^{\mathsf{C}}} \subseteq R^{\mathsf{C}}$ and $R^{\mathsf{C}}\{\overline{R^{\mathsf{C}}}\} \subseteq R^{\mathsf{C}}$. By induction hypothesis $q' \Downarrow v'$ with $\vdash v \ \overline{R^{\mathsf{C}}} \ v'$ and we conclude $p' \Downarrow v'$ by (Eval case).

We conclude (5.13), so R^{C} is an open simulation contained in $\lesssim^{\circ}$. □

Proposition 4 is not a complete proof rule. For example, $\texttt{fn}\ x \texttt{ => } x$ and $\texttt{fn}\ x \texttt{ => I}\ x$ are bisimilar but they are not related by any closed relation R such that $x \vdash x \ R^{\mathsf{C}} \ \texttt{I}\ x$, because x and $\texttt{I}\ x$ have no common context and R relates only closed expressions.

A more satisfactory and complete rule for simulation up to context would be

$$\frac{R \subseteq \langle (R^{\circ})^{\mathsf{C}} \rangle}{R \subseteq \lesssim} \tag{5.14}$$

Unfortunately, our attempts to prove (or refute) this have failed and we leave it as an open problem. A simple calculation, using (3.2), shows that (5.14) is equivalent to the 'open' rule:

$$\frac{R \subseteq \langle R^{\mathsf{C}} \rangle^{\circ}}{R \subseteq \lesssim^{\circ}}$$

It differs from Proposition 4 in that R may be open so that R^{C} may capture free variables in expressions related by R. The premise is equivalent to $R\{Id\} \subseteq \langle R^{\mathsf{C}} \rangle$, by (3.2). In Section 7 we prove a weaker version where we require that the premise holds not only for identical instantiations, $R\{Id\}$, but also for $\overline{R^{\mathsf{C}}}$ related instantiations, $R\{\overline{R^{\mathsf{C}}}\}$,

$$\frac{R\{\overline{R^{\mathsf{C}}}\} \cap Rel \subseteq \langle R^{\mathsf{C}} \rangle}{R \subseteq \lesssim^{\circ}} \tag{5.15}$$

If R is closed then $R\{\overline{R^{\mathsf{C}}}\} = R$ and (5.15) reduces to Proposition 4.

We have found neither simulation up to $\lesssim$ nor up to context to be particularly useful in themselves. It seems that their potential is only realised when combined. In connection with the precongruence proof for similarity below, we will show a stronger version of Proposition 4.

Proposition 5 (Simulation up to context and $\lesssim$) $\dfrac{R \subseteq \langle R^{\mathsf{C}} \lesssim^{\circ} \rangle}{R \subseteq \lesssim}$

This can be used to show that the `Yv` fixed point combinator enjoys a least pre-fixed point induction rule:

Proposition 6 `fn` x `=>` $e\{^{v}\!/\!_{f}\} \lesssim v$ *implies* `Yv` (`fn` f `=>` `fn` x `=>` $e) \lesssim v$.

Proof Let $u \stackrel{\text{def}}{=}$ `fn` f `=>` `fn` x `=>` e, $w \stackrel{\text{def}}{=}$ `fn` x `=>` $u\ u^{\infty}\ x$. Recall $u^{\infty} =$ (`fn` f `=>` `fn` x `=>` $u(f\,f)x$) (`fn` f `=>` `fn` x `=>` $u(f\,f)x$). We have `Yv` $u \Downarrow_2 w$, $u^{\infty} \Downarrow_1 w$, and `Yv` $u \sim u^{\infty}$.

Now assume the premise holds. It suffices to show $u^{\infty} \lesssim v$. We proceed by simulation up to context and $\lesssim$. Let $R \stackrel{\text{def}}{=} \{(u^{\infty}, v)\}$, then $R \subseteq \langle R^{\mathsf{C}} \lesssim^{\circ} \rangle$ holds because $u^{\infty} \Downarrow w =$ `fn` x `=>` $u\ u^{\infty}\ x$ and we have

$$\begin{array}{lll} \vdash \texttt{fn}\ x \texttt{ => } u\ u^{\infty}\ x & \overline{R^{\mathsf{C}}} & \texttt{fn}\ x \texttt{ => } u\ v\ x \\ & \overline{\sim^{\circ}} & \texttt{fn}\ x \texttt{ => } e\{^{v}\!/\!_{f}\} \\ & \overline{\lesssim^{\circ}} & v, \end{array}$$

by Kleene equivalence, assumption, and (5.5). Bisimulation up to context and $\lesssim$ yields $R \subseteq \lesssim$. Hence $u^{\infty} \lesssim v$ and we conclude `Yv` $u \lesssim v$, as required. $\square$

For example, by the fixed point law for `rec`, we get

$$\texttt{Yv}\,(\texttt{fn}\ f\ \texttt{=>}\ \texttt{fn}\ x\ \texttt{=>}\ e)\ \lesssim\ \texttt{rec}\ f\ x\ \texttt{=>}\ e. \tag{5.16}$$

Proposition 5 is still too restrictive for many purposes. For instance, it is not clear how to prove the converse of (5.16) and the least pre-fixed point induction rule for `rec`. One might expect simulation up to $\lesssim$ and simulation up to context to combine as follows:

$$\frac{R \subseteq \langle \lesssim^{\circ}\ R^{\mathsf{C}}\ \lesssim^{\circ} \rangle}{R \subseteq \lesssim} \tag{5.17}$$

But this fails. As a counterexample (due to Andrew Gordon) take

$$R\ \stackrel{\text{def}}{=}\ \{(\texttt{fn}\ x\ \texttt{=>}\ \texttt{nil}, \texttt{fn}\ x\ \texttt{=>}\ \Omega)\}. \tag{5.18}$$

Clearly $x \vdash \texttt{nil} \not\lesssim^{\circ} \Omega$ and $\texttt{fn}\ x\ \texttt{=>}\ \texttt{nil} \not\lesssim \texttt{fn}\ x\ \texttt{=>}\ \Omega$. But $R \subseteq \langle \sim^{\circ}\ R^{\mathsf{C}}\ \sim^{\circ} \rangle$ because

$$x \vdash \texttt{nil} \sim^{\circ} (\texttt{fn}\ x\ \texttt{=>}\ \texttt{nil})\,\texttt{nil}\ R^{\mathsf{C}}\ (\texttt{fn}\ x\ \texttt{=>}\ \Omega)\,\texttt{nil} \sim^{\circ} \Omega. \tag{5.19}$$

This failure corresponds to the situation for process calculi, where a symmetric rule for weak bisimulation up to context and weak bisimulation also fails (Sangiorgi 1996). There the rule is repaired by introducing a more fine-grained efficiency preorder, called expansion. In Section 6 we develop a corresponding improvement relation for our language. Then we repair (5.17) by replacing the left occurrence of similarity in the premise with improvement (Proposition 10).

5.2 Precongruence

A *precongruence* is a compatible preorder, that is, a preorder which is preserved by all language constructs. Precongruence is an important property of similarity because it allows compositional (in)equational reasoning. Moreover, it shows that bisimilarity coincides with conventional contextual equivalence (an issue which we shall not address in this paper, however). We shall now prove that similarity is a precongruence by means of Howe's general method for proving congruence of simulation orderings (Howe 1996). We employ an extension of the method, due to Pitts (1995), which also establishes the simulation up to context results of the previous section.

Recall that $\lesssim^{\circ}$ is a preorder. It is a precongruence if it is also compatible, $\widehat{\lesssim^{\circ}} \subseteq \lesssim^{\circ}$. Howe proves this by simulation but since $\widehat{\lesssim^{\circ}}$ is not itself a simulation he constructs a larger 'candidate relation' which is. Pitts parameterises the candidate relation by a closed relation R. For every $R \subseteq Rel$, the parameterised candidate relation, $R^{\sharp}$, is defined inductively by

$$(\text{Cand } R)\ \frac{p\ R\ p' \quad \vec{x} \vdash p' \lesssim^{\circ} e''}{\vec{x} \vdash p\ R^{\sharp}\ e''} \qquad (\text{Cand Comp})\ \frac{\vec{x} \vdash e\ \widehat{R^{\sharp}}\ e' \quad \vec{x} \vdash e' \lesssim^{\circ} e''}{\vec{x} \vdash e\ R^{\sharp}\ e''}$$

Each $R^{\sharp}$ satisfies all the properties of Howe's candidate relation:

Lemma 4 *(1) $R^\sharp$ is reflexive, compatible, and substitutive.*

(2) $R^\sharp$ contains similarity, $\lesssim^\circ \subseteq R^\sharp$.

(3) $R^\sharp$ contains its composition with similarity, $R^\sharp \lesssim^\circ \subseteq R^\sharp$.

Proof Compatibility, $\widehat{R^\sharp} \subseteq R^\sharp$, is immediate from (Cand Comp) because $\lesssim^\circ$ is reflexive. Every compatible relation is reflexive, so $R^\sharp$ and $\widehat{R^\sharp}$ are reflexive. Again by (Cand Comp) follows (2). Since $\lesssim^\circ$ is transitive, (3) is also immediate from the definition of $R^\sharp$. Weakening,

$$\vec{x}\vec{y} \vdash e \; R^\sharp \; e' \quad \Rightarrow \quad \vec{x}x\vec{y} \vdash e \; R^\sharp \; e',$$

can be shown by induction on the derivation of $\vec{x}\vec{y} \vdash e \; R^\sharp \; e'$, using the fact that $\lesssim^\circ$ satisfies weakening, being an open extension. Finally, substitutivity,

$$\vec{x} \vdash e \; R^\sharp \; e' \;\; \& \;\; \vec{y} \vdash \vec{u} \; R^\sharp \; \vec{u}' \quad \Rightarrow \quad \vec{y} \vdash e\{\vec{u}/\vec{x}\} \; R^\sharp \; e'\{\vec{u}'/\vec{x}\},$$

is proved by induction on the derivation of $\vec{x} \vdash e \; R^\sharp \; e'$, using (3), weakening, and the fact that $\lesssim^\circ$ is closed under substitutions. $\Box$

Lemma 5 $\dfrac{R \subseteq \langle R^\sharp \rangle}{R^\sharp \subseteq \lesssim^\circ}$

Proof As in the proof of Proposition 4, it suffices to prove $R^\sharp \cap Rel \subseteq \langle R^\sharp \rangle$, then $R^\sharp$ will be an open simulation and $R^\sharp \subseteq \lesssim^\circ$.

We assume $\vdash p \; R^\sharp \; p''$ and $p \Downarrow v$, and we will prove that there exists v'' such that $p'' \Downarrow v''$ and $\vdash v \; \overline{R^\sharp} \; v''$. The proof is by induction on the derivation of $p \Downarrow v$.

First consider the derivation of $\vdash p \; R^\sharp \; p''$. We see that there exists p' such that $p' \lesssim p''$ and either $p \; R \; p'$ or $p \; \widehat{R^\sharp} \; p'$. In either case we argue exactly as in the proof of Proposition 4 to get that $p' \Downarrow v'$, for some v' such that $\vdash v \; \overline{R^\sharp} \; v'$. Then, by definition of $\lesssim$, there exists v'' such that $p'' \Downarrow v''$ and $\vdash v' \; \overline{\lesssim^\circ} \; v''$. Since $\overline{R^\sharp} \; \overline{\lesssim^\circ} \subseteq \overline{R^\sharp \lesssim^\circ}$ and $R^\sharp \lesssim^\circ \subseteq R^\sharp$, $\vdash v \; \overline{R^\sharp} \; v''$ follows, as required. $\Box$

Howe's candidate relation is just $\emptyset^\sharp$, for which the premise of the lemma holds trivially and thus establishes $\emptyset^\sharp \subseteq \lesssim^\circ$. We have the reverse inclusion from above, so $\lesssim^\circ$ and $\emptyset^\sharp$ coincide. Since $\emptyset^\sharp$ is substitutive and compatible, so is $\lesssim^\circ$. As $\lesssim^\circ$ is also a preorder, it is a precongruence.

Proposition 7 $\lesssim^\circ$ *is substitutive and a precongruence.*

Consequently, $\sim^\circ$ is also substitutive and is a congruence, that is, a compatible equivalence relation.

Lemma 5 also entails Proposition 4: since $\lesssim^\circ$ is reflexive we see that $R^{\mathsf{C}} \subseteq R^\sharp$; therefore $R \subseteq \langle R^{\mathsf{C}} \rangle$ implies $R \subseteq \langle R^\sharp \rangle$ and Lemma 5 gives $R^\sharp \subseteq \lesssim^\circ$ and thus $R \subseteq \lesssim$. Moreover, since $R^\sharp \lesssim^\circ \subseteq R^\sharp$, also $R^{\mathsf{C}} \lesssim^\circ \subseteq R^\sharp$, so Lemma 5 entails Proposition 5 too.

5.3 Equational theory

Let us summarise our results about bisimilarity from above. We supplement some equational laws that follow directly from (5.1)–(5.4) by inspection of evaluations.

Extensionality

$$v \sim v' \quad \text{iff} \quad \vdash v \ \overline{\sim^\circ}\ v' \quad \text{iff} \quad \vdash v \ \widehat{\sim^\circ}\ v'.$$

$$\vec{x} \vdash e \sim^\circ e' \quad \text{iff} \quad \forall u_1, \ldots, u_n \in \mathit{Val}_\emptyset.\ e\{\vec{u}/\vec{x}\} \sim e'\{\vec{u}/\vec{x}\}.$$

The latter is just the definition of open extension.

Congruence and substitutivity

$$p \sim p.$$

$$p \sim p' \quad \text{and} \quad p' \sim p'' \quad \text{imply} \quad p \sim p''.$$

$$p \sim p' \quad \text{implies} \quad p' \sim p.$$

$$\vdash p \ \widehat{\sim^\circ}\ p' \quad \text{implies} \quad p \sim p'.$$

$$x \vdash e \sim^\circ e' \quad \text{and} \quad v \sim v' \quad \text{imply} \quad e\{v/x\} \sim e'\{v'/x\}.$$

Strachey's property (Gordon 1994)

$$\text{either} \quad p \sim \Omega \quad \text{or} \quad \exists v.\, p \sim v.$$

Functions

$$(\texttt{fn}\ x\ \texttt{=>}\ e)\ v \ \sim\ e\{v/x\}.$$

Combined with the extensionality laws we get:

$$\texttt{fn}\ x\ \texttt{=>}\ e \sim \texttt{fn}\ x\ \texttt{=>}\ e' \quad \text{iff} \quad \forall v.\, (\texttt{fn}\ x\ \texttt{=>}\ e)\ v \sim (\texttt{fn}\ x\ \texttt{=>}\ e')\ v.$$

Let We have laws corresponding to those of Moggi's computational lambda calculus (Moggi 1989), here presented as in Talcott (1997).

$$\texttt{let val}\ x = v\ \texttt{in}\ e\ \texttt{end} \ \sim\ e\{v/x\}.$$

$$\texttt{let val}\ x = p\ \texttt{in}\ E[x]\ \texttt{end} \ \sim\ E[p],$$

where E is any closed evaluation context.

Actually, these laws follow from the laws for `case` below because `let` can be encoded by means of `case`,

$$\begin{pmatrix} \texttt{let val}\ x = p \\ \texttt{in}\ e\ \texttt{end} \end{pmatrix} \sim \begin{pmatrix} \texttt{case}\ p\ \texttt{of nil =>}\ e\{\texttt{nil}/x\} \\ \qquad \texttt{|}\ x_1\ \texttt{::}\ x_2\ \texttt{=>}\ e\{x_1\ \texttt{::}\ x_2/x\} \\ \qquad \texttt{|}\ f\ \texttt{=>}\ e\{f/x\} \end{pmatrix}.$$

Case

$$\begin{pmatrix}\texttt{case}\ v\ \texttt{of nil =>}\ e_1 \\ \qquad\ \texttt{|}\ x_1\ \texttt{::}\ x_2\ \texttt{=>}\ e_2 \\ \qquad\ \texttt{|}\ f\ \texttt{=>}\ e_3\end{pmatrix} \sim \begin{cases} e_1 & \text{if } v = \texttt{nil} \\ e_2\{v_1/x_1, v_2/x_2\} & \text{if } v = v_1\ \texttt{::}\ v_2 \\ e_3\{v/f\} & \text{if } v = \texttt{fn}\ x\ \texttt{=>}\ e. \end{cases}$$

$$\begin{pmatrix}\texttt{case}\ p\ \texttt{of nil =>}\ E[\texttt{nil}] \\ \qquad\ \texttt{|}\ x_1\ \texttt{::}\ x_2\ \texttt{=>}\ E[x_1\ \texttt{::}\ x_2] \\ \qquad\ \texttt{|}\ f\ \texttt{=>}\ E[f]\end{pmatrix} \sim E[p].$$

Fixed point `rec` is a fixed point operator:

$$\texttt{rec}\ f\ x\ \texttt{=>}\ d \ \sim\ \texttt{fn}\ x\ \texttt{=>}\ d\{(\texttt{rec}\ f\ x\ \texttt{=>}\ d)/f\}.$$

Furthermore, `rec` is rationally open (Braüner 1996):

$$C[\texttt{rec}\ f\ x\ \texttt{=>}\ d] \ \sim\ \Omega \quad \text{iff} \quad \forall n \geq 0.\ C[\texttt{rec}^{(n)}\ f\ x\ \texttt{=>}\ d] \ \sim\ \Omega.$$

This rule is useful for equational reasoning about divergence, without direct reference to the evaluation relation. Rational openness and the unwinding theorem are easily derived from each other using adequacy (5.7). (Rational openness also follows from syntactic continuity and syntactic bottom below.)

5.4 Inequational theory

We also list some order-theoretic properties of similarity, taken from Pitts (1997), in order to complete our discussion about proof rules for recursion.

Extensionality, precongruence, and substitutivity As for $\sim$ above, except symmetry.

Syntactic bottom Ω is least with respect to $\lesssim$,

$$\Omega \lesssim p.$$

This is direct from (5.1).

Recursion induction `rec` f x `=>` e is the least pre-fixed point of the functional `fn` x `=>` $e\{-/f\}$,

$$\texttt{fn}\ x\ \texttt{=>}\ e\{v/f\} \ \lesssim\ v \quad \text{implies} \quad \texttt{rec}\ f\ x\ \texttt{=>}\ e \ \lesssim\ v.$$

In Section 5.1 we proved this result for the `Yv` combinator, Proposition 6. The recursion induction rule for `rec` follows from syntactic continuity below; see Pitts (1997). In the following sections we shall discuss other proofs of recursion induction using improvement and simulation up to context.

Syntactic continuity Every recursive function is the least upper bound of its finite approximants and all language constructs are continuous with respect to this least upper bound,

$$C[\texttt{rec}\ f\ x\ \texttt{=>}\ e]\ \lesssim\ q \quad \text{iff} \quad \forall n \geq 0.\ C[\texttt{rec}^{(n)}\ f\ x\ \texttt{=>}\ e]\ \lesssim\ q.$$

Proof We employ Lemma 3 from the proof of the unwinding theorem in Section 4 to give a co-inductive proof. A similar proof is outlined in Pitts (1997). Here we can use the relations from the formulation of Lemma 3 to construct the appropriate simulations. Note that we do not require that $\texttt{rec}\ f\ x\ \texttt{=>}\ e$ is closed.

First consider the backward implication (which is the most interesting). Recall the relations U_n from Lemma 3(1). We construct the relation

$$T \ \stackrel{\text{def}}{=}\ \bigcap_{n \geq 0} (U_n \lesssim^\circ).$$

Observe that $\forall n \geq 0.\ C[\texttt{rec}^{(n)}\ f\ x\ \texttt{=>}\ e] \lesssim q$ implies $\vdash C[\texttt{rec}\ f\ x\ \texttt{=>}\ e]\ T\ q$. We show that T is an open simulation, then $T \subseteq \lesssim^\circ$ and the result follows. So suppose $\vdash p\ T\ p'$ and $p \Downarrow_N v$. By definition of T, for all $n \geq 0$, $\vdash p\ U_{n+1+N}\ p_n$, for some $p_n \lesssim p'$. From Lemma 3(1) we get $p_n \Downarrow_N v_n$ with $\vdash v\ U_{n+1}\ v_n$. By the same argument as for (4.2) holds

$$\vdash u\ U_{n+1}\ u' \quad \text{implies} \quad \vdash u\ \overline{U_n}\ u', \tag{5.20}$$

so we get $\vdash\ v\ \overline{U_n}\ v_n$. Since $p_n\ \lesssim\ p'$ also $p' \Downarrow v_n'$ with $\vdash\ v_n\ \overline{\lesssim^\circ}\ v_n'$, hence $\vdash\ v\ \overline{U_n \lesssim^\circ}\ v_n'$. Evaluation is deterministic so all v_n' are identical. Therefore $\vdash v\ \overline{U_n \lesssim^\circ}\ v_0'$, for all $n \geq 0$, and we conclude $\vdash\ v\ \overline{T}\ v_0'$ and $p\ \langle T \rangle\ p'$, hence T is an open simulation as it is closed under substitutions.

The forward implication holds if $C[\texttt{rec}^{(n)}\ f\ x\ \texttt{=>}\ e] \lesssim C[\texttt{rec}\ f\ x\ \texttt{=>}\ e]$ for all n. This is derivable from syntactic bottom and precongruence, by induction on n. It also follows from Lemma 3(2). To see this we first extend (5.20) with

$$\vdash v\ U_0^{\text{op}}\ v' \quad \text{implies} \quad \vdash v\ \overline{U_0^{\text{op}} \cup \lesssim^\circ}\ v'. \tag{5.21}$$

This holds because $\vdash (\texttt{rec}^{(0)}\ f\ x\ \texttt{=>}\ e)\ \overline{\lesssim^\circ}\ (\texttt{rec}\ f\ x\ \texttt{=>}\ e)$, by syntactic bottom. Lemma 3(2) gives $U_0^{\text{op}} \cap \mathit{Rel} \subseteq \langle U_0^{\text{op}} \cup \lesssim^\circ \rangle$ and we deduce $U_0^{\text{op}} \cup \lesssim^\circ$ is an open simulation and $U_0^{\text{op}} \subseteq \lesssim^\circ$. From (4.1) we get

$$C[\texttt{rec}^{(n)}\ f\ x\ \texttt{=>}\ e] \lesssim C[\texttt{rec}\ f\ x\ \texttt{=>}\ e],$$

as required. $\square$

Determinacy of evaluation plays a key role in the above proof of syntactic continuity. One can add nondeterminism to the language such that the operational semantics and theory of applicative bisimulation still satisfy the unwinding theorem, rational openness, and recursion induction, but syntactic continuity fails. Braüner (1996) uses this example to illustrate that syntactic continuity is a strictly stronger property than rational openness.

6 Improvement

Following Sands (1997b) we introduce a stricter operational ordering and equivalence that takes computational cost into account, in our case the number of function applications in evaluations. Improvement theory has independent interest as a formal approach to the study of program efficiency but Sands has also demonstrated that it is a powerful tool for reasoning about conventional operational equivalence and recursion. Here we are interested in the latter use of improvement.

We study the theory in some detail as its scope goes far beyond repairing the rule for simulation up to context and $\lesssim$ of the previous section. Our relational approach is instrumental in establishing a rule for improvement simulation up to variable capturing contexts. This is interesting in its own right, especially in the absence of a satisfactory counterpart for similarity, and it entails Sands' improvement theorem.

As motivation for our definition of improvement below, recall that (5.19),

$$x \vdash \texttt{nil} \sim^\circ (\texttt{fn}\; x \;\texttt{=>}\; \texttt{nil})\,\texttt{nil}\; R^{\mathsf{C}}\; (\texttt{fn}\; x \;\texttt{=>}\; \Omega)\,\texttt{nil} \sim^\circ \Omega,$$

was used to prove $R \subseteq \langle \lesssim^\circ\; R^{\mathsf{C}}\; \lesssim^\circ \rangle$ and thus invalidated the symmetric up to context and $\lesssim$ rule (5.17),

$$\frac{R \subseteq \langle \lesssim^\circ\; R^{\mathsf{C}}\; \lesssim^\circ \rangle}{R \subseteq \lesssim}$$

Here `nil` is bisimilar to $(\texttt{fn}\; x \;\texttt{=>}\; \texttt{nil})\,\texttt{nil}$ but the latter is more "expensive" as it takes one more function application step to compute. So `nil` is not "improved" by $(\texttt{fn}\; x \;\texttt{=>}\; \texttt{nil})\,\texttt{nil}$. This will be the requirement by which we shall repair (5.17) in Section 7.

We measure the number of function applications in evaluations, essentially because applications 'destruct' function abstractions. In fact, the counterexample to (5.17) can be constructed with any 'lazy' value constructor and associated destructors, but function abstraction and application happen to be the only lazy value constructor and destructor in ML. In general the cost measure must count every destruction of any lazy constructor. We should mention that this is tailored to support reasoning about applicative similarity and it is not meant as a contribution to the discussion of what constitutes a good measure of program effeciency for functional languages (Lawall and Mairson 1996).

We define an improvement preorder, $\trianglerighteq\!\!\!\sim$, and a cost equivalence relation, $\diamond\!\!\!\sim$, co-inductively like similarity and bisimilarity but with the additional requirement that $p \trianglerighteq\!\!\!\sim q$ implies that q evaluates in less function application steps than p. The definitions and the basic theory are quite analogous to those of Section 4.

Let the improvement simulation operator, $\langle \_ \rangle_I$, be given by

$$p \,\langle R \rangle_I\, p' \;\stackrel{\text{def}}{\Leftrightarrow}\; \forall N.\forall v.\, p \Downarrow_N v \;\;\Rightarrow\;\; \exists v'.\, p' \Downarrow_{\leq N} v' \;\;\&\;\; \vdash v\, \overline{R}\, v',$$

for $R \subseteq \mathit{Rel}^\circ$, $p, p' \in \mathit{Exp}_\emptyset$. Notation $p' \Downarrow_{\leq N} v'$ means $\exists N' \leq N.\; p' \Downarrow_{N'} v'$. The compound operator $\langle \_^\circ \rangle_I$ is monotone and we define *improvement*, $\trianglerighteq\!\!\!\sim$, as the greatest fixed point

$$\trianglerighteq \stackrel{\text{def}}{=} \nu R.\langle R^{\circ}\rangle_I, \tag{6.1}$$

Cost equivalence, $\trianglelefteq\!\trianglerighteq$, is the greatest symmetric fixed point and is also the largest symmetric relation contained in improvement,

$$\trianglelefteq\!\trianglerighteq \;=\; \trianglerighteq \cap \trianglerighteq^{\mathrm{op}}.$$

Cost equivalence is computationally adequate, (5.7). But the evaluation relation is not sound, (5.9), with respect to cost equivalence; instead we have a more detailed correspondence between evaluation and cost equivalence:

$$p \Downarrow_N v \quad \text{implies} \quad p \trianglelefteq\!\trianglerighteq \mathtt{I}^N v.$$

Application of the identity function $\mathtt{I}$ is used as syntactic representation of function application steps. $\mathtt{I}^N v$ means N-fold application of $\mathtt{I}$ to v.

We call post-fixed points of $\langle \_^{\circ}\rangle_I$ *improvement simulations* and we have co-induction improvement simulation rules:

$$\frac{R \subseteq \langle R^{\circ}\rangle_I}{R \subseteq \trianglerighteq} \qquad \frac{R \subseteq \langle R\rangle_I^{\circ}}{R \subseteq \trianglerighteq^{\circ}} \tag{6.2}$$

We call R an *improvement simulation* if $R \subseteq \langle R^{\circ}\rangle_I$ and R is an *open improvement simulation* if $R \subseteq \langle R\rangle_I^{\circ}$.

Improvement refines similarity, $\trianglerighteq \subseteq \lesssim$, because

$$\langle R\rangle_I \subseteq \langle R\rangle, \text{ for all } R \subseteq Rel^{\circ}, \tag{6.3}$$

so every improvement simulation is also an (applicative) simulation.

6.1 Improvement simulation up to context

Refined simulation rules are equally important for improvement as they are for applicative simulation. It turns out that we are able to prove stronger refinements of improvement simulation than was the case for applicative simulation. In the process we will also derive that improvement is a precongruence.

Lemma 6 $\dfrac{R \subseteq \langle S^{+}\rangle_I}{S^{+} \subseteq \trianglerighteq^{\circ}}$, *where* $S \stackrel{\text{def}}{=} (R^{\circ})^{\mathsf{C}}$ *and* S^{+} *is the transitive closure.*

The proof is postponed to Appendix A.

Proposition 8 $\trianglerighteq^{\circ}$ *is substitutive and a precongruence.*

Proof As $\underset{\sim}{\triangleright}$ itself satisfies the premise of the lemma, we get $(\underset{\sim}{\triangleright}^{\circ})^{\mathsf{C}+} \subseteq \underset{\sim}{\triangleright}^{\circ}$. Therefore $\underset{\sim}{\triangleright}^{\circ}$ is compatible and transitive, and hence a precongruence. Since $\underset{\sim}{\triangleright}^{\circ}$ is closed under substitutions, it is also substitutive by Lemma 1. $\square$

Another consequence of Lemma 6 is a full symmetric rule for improvement simulation up to context and improvement.

Proposition 9 (Improvement simulation up to context and $\underset{\sim}{\triangleright}$)

$$\frac{R \subseteq \langle \underset{\sim}{\triangleright}^{\circ} \, (R^{\circ})^{\mathsf{C}} \, \underset{\sim}{\triangleright}^{\circ} \rangle_I}{R \subseteq \underset{\sim}{\triangleright}}$$

Proof From $R \subseteq \langle \underset{\sim}{\triangleright}^{\circ} \, (R^{\circ})^{\mathsf{C}} \, \underset{\sim}{\triangleright}^{\circ} \rangle_I$ we get that $R \cup \underset{\sim}{\triangleright} \subseteq \langle ((R \cup \underset{\sim}{\triangleright})^{\circ})^{\mathsf{C}+} \rangle_I$. Hence $((R \cup \underset{\sim}{\triangleright})^{\circ})^{\mathsf{C}+} \subseteq \underset{\sim}{\triangleright}^{\circ}$ and $R \subseteq \underset{\sim}{\triangleright}$. $\square$

6.2 Equational theory

The equational theory of cost equivalence is analogous to that of bisimilarity, except for some applications of the identity function, `I`, to account for computational cost. Since these 'syntactic computation steps' can be erased up to bisimilarity, the cost equivalence theory here entails the corresponding theory of bisimilarity in Section 4.

The cost equivalence version of Strachey's property accounts for the cost of computing a value:

$$\text{either} \quad p \underset{\sim}{\lhd\!\rhd} \Omega \quad \text{or} \quad \exists! N. \exists v.\, p \underset{\sim}{\lhd\!\rhd} \mathtt{I}^N v.$$

The beta law for function application records the computation step:

$$(\mathtt{fn}\ x \ \mathtt{=>}\ e)\ v \ \underset{\sim}{\lhd\!\rhd} \ \mathtt{I}\ e\{v/x\}.$$

Notice that $\mathtt{let}\ \mathtt{val}\ x = d\ \mathtt{in}\ e\ \mathtt{end}$ is one step 'cheaper' than the conventional encoding $(\mathtt{fn}\ x \ \mathtt{=>}\ e)\ d$. This will be important in the proof of Proposition 11.

The remaining equational laws for bisimilarity in Section 5.3 carry over to cost equivalence unchanged.

The laws can be used to move around syntactic computation steps. For instance, the evaluation context law for `case` yields

$$\mathtt{I}\begin{pmatrix} \mathtt{case}\ p\ \mathtt{of}\ \mathtt{nil}\ \mathtt{=>}\ e_1 \\ \quad\quad \mid x_1\ \mathtt{::}\ x_2\ \mathtt{=>}\ e_2 \\ \quad\quad \mid f\ \mathtt{=>}\ e_3 \end{pmatrix} \sim \begin{array}{l} \mathtt{case}\ p\ \mathtt{of}\ \mathtt{nil}\ \mathtt{=>}\ \mathtt{I}\ e_1 \\ \quad\quad \mid x_1\ \mathtt{::}\ x_2\ \mathtt{=>}\ \mathtt{I}\ e_2 \\ \quad\quad \mid f\ \mathtt{=>}\ \mathtt{I}\ e_3, \end{array}$$

because $(\mathtt{I}-)$ is an evaluation context. A further law of this kind,

$$E[\mathtt{I}\ p] \ \underset{\sim}{\lhd\!\rhd} \ \mathtt{I}\ E[p],$$

moves `I` across evaluation contexts; it is direct from (2.2). Such laws form a useful 'tick algebra' (Sands 1997b) for equational reasoning about computation steps.

Cost equivalence satisfies a unique fixed point rule:

$$\texttt{fn}\ x\ \texttt{=>}\ e\{v/f\} \mathrel{\underset{\sim}{\lhd\!\rhd}} v \quad \text{implies} \quad \texttt{rec}\ f\ x\ \texttt{=>}\ e \mathrel{\underset{\sim}{\lhd\!\rhd}} v.$$

This rule follows from recursion induction and co-induction rules below. For illustration, we can use it to prove the following correspondence between explicit recursion and the `Yv` fixed point combinator.

$$\texttt{I}^2\ (\texttt{rec}\ f\ x\ \texttt{=>}\ \texttt{I}^3\, e) \mathrel{\underset{\sim}{\lhd\!\rhd}} \texttt{Yv}\ u, \quad \text{where } u = (\texttt{fn}\ f\ \texttt{=>}\ \texttt{fn}\ x\ \texttt{=>}\ e),$$

by calculating $\texttt{Yv}\ u \mathrel{\underset{\sim}{\lhd\!\rhd}} \texttt{I}^2\ \texttt{fn}\ x\ \texttt{=>}\ u\ u^\infty\, x$ and $x \vdash \texttt{I}^3\, e\{(\texttt{fn}\ x\ \texttt{=>}\ u\ u^\infty\ x)/f\} \mathrel{\underset{\sim}{\lhd\!\rhd}}^\circ u\ u^\infty\, x$. As usual, a corresponding result for bisimilarity, $\texttt{rec}\ f\ x\ \texttt{=>}\ e \sim \texttt{Yv}\ u$, follows as a corollary. This and Proposition 6 constitute a proof of recursion induction for similarity.

6.3 Inequational theory

All the inequational theory for similarity in Section 5.4 also holds for improvement.

The proofs of syntactic bottom and syntactic continuity for improvement are again analogous to those for similarity above. The lemmas from the proof of the unwinding theorem in Section 4 were carefully phrased to also account for computational cost and the syntactic continuity proof for similarity is easily extended with this bookkeeping.

We supplement recursion induction,

$$\texttt{fn}\ x\ \texttt{=>}\ e\{v/f\} \mathrel{\underset{\sim}{\rhd}} v \quad \text{implies} \quad \texttt{rec}\ f\ x\ \texttt{=>}\ e \mathrel{\underset{\sim}{\rhd}} v,$$

with recursion co-induction,

$$v \mathrel{\underset{\sim}{\rhd}} \texttt{fn}\ x\ \texttt{=>}\ e\{v/f\} \quad \text{implies} \quad v \mathrel{\underset{\sim}{\rhd}} \texttt{rec}\ f\ x\ \texttt{=>}\ e,$$

which says that recursive functions are also greatest post-fixed points with respect to improvement. We can use improvement simulation up to context and $\underset{\sim}{\rhd}$ to prove the recursion (co-)induction rules.

Proof of recursion (co-)induction We only prove the first (induction) rule. The second (co-induction) rule follows by a symmetric argument because the improvement simulation up to context and $\underset{\sim}{\rhd}$ rule, Proposition 9, is symmetric.

Assume $\texttt{fn}\ x\ \texttt{=>}\ e\{v/f\} \mathrel{\underset{\sim}{\rhd}} v$. By extensionality, $\vdash \texttt{fn}\ x\ \texttt{=>}\ e\{v/f\} \mathrel{\overline{\underset{\sim}{\rhd}^\circ}} v$. Let R be the singleton relation, $r\ R\ v$, where $r = \texttt{rec}\ f\ x\ \texttt{=>}\ e$. Then r is a fixed point, $r \mathrel{\underset{\sim}{\lhd\!\rhd}} \texttt{fn}\ x\ \texttt{=>}\ e\{r/f\}$, and $\vdash r \mathrel{\overline{\underset{\sim}{\lhd\!\rhd}^\circ}} \texttt{fn}\ x\ \texttt{=>}\ e\{r/f\}$. Now $r\ \langle \underset{\sim}{\lhd\!\rhd}^\circ\ (R^\circ)^{\mathsf{C}}\ \underset{\sim}{\rhd}^\circ \rangle_I\ v$ because $r \Downarrow_0 r$, $v \Downarrow_0 v$, and

$$\vdash r \mathrel{\overline{\underset{\sim}{\lhd\!\rhd}^\circ}} \texttt{fn}\ x\ \texttt{=>}\ e\{r/f\} \mathrel{\overline{(R^\circ)^{\mathsf{C}}}} \texttt{fn}\ x\ \texttt{=>}\ e\{v/f\} \mathrel{\overline{\underset{\sim}{\rhd}^\circ}} v.$$

Hence $R \subseteq \langle \underset{\sim}{\triangleleft\!\triangleright}^{\circ}\ (R^{\circ})^{\mathsf{C}}\ \underset{\sim}{\triangleright}^{\circ} \rangle_{I}$. By Proposition 9, we conclude $R \subseteq \underset{\sim}{\triangleright}$, i.e., $r \underset{\sim}{\triangleright} v$. □

Recursion co-induction and the unique fixed point rule are call-by-value versions of Sands' improvement theorem. This is apparent from the following reformulation, derived by means of equational laws for $\underset{\sim}{\triangleleft\!\triangleright}$.

$$\begin{array}{rl} x \vdash \texttt{let fun}\ f\ x = d_0\ \texttt{in}\ d_0\ \texttt{end}\ \underset{\sim}{\triangleright}^{\circ}\ \texttt{let fun}\ f\ x = d_0\ \texttt{in}\ d_1\ \texttt{end} & \\ \Rightarrow\ \ \texttt{let fun}\ f\ x = d_0\ \texttt{in}\ e\ \texttt{end}\ \underset{\sim}{\triangleright}\ \texttt{let fun}\ f\ x = d_1\ \texttt{in}\ e\ \texttt{end}, & \end{array}$$

for $d_0, d_1 \in \mathit{Exp}_{fx}$ and $e \in \mathit{Exp}_{f}$. The same holds for $\underset{\sim}{\triangleleft\!\triangleright}$.

It should be noted that a reason why our improvement theory satisfies the improvement theorem is that recursion is bound up with function abstraction in ML, that is, recursive unfoldings require a function application step (cf. the general version of the improvement theorem in Sands (1997a)). Hence our cost measure is actually more fine-grained than Sands' count of unfoldings of recursion in Sands (1997b). In languages where recursion is not coupled with function abstraction, the two cost measures are incomparable and the two resulting improvement theories will be complementary.

7 Applicative simulation up to improvement

A motivation for introducing improvement is its use in refining applicative simulation. We can extend Proposition 5 as follows.

Proposition 10 (Applicative simulation up to $\underset{\sim}{\triangleright}$ and context and $\lesssim$)

$$\frac{R \subseteq \langle \underset{\sim}{\triangleright}^{\circ}\ R^{\mathsf{C}}\ \lesssim^{\circ} \rangle}{R \subseteq \lesssim}$$

This rule allows us to give a direct proof of recursion induction for similarity, analogous to the proof for improvement above: suppose $\texttt{fn}\ x\ \texttt{=>}\ e\{v/f\} \lesssim v$ and let $R = \{(\texttt{rec}\ f\ x\ \texttt{=>}\ e, v)\}$, then

$$\texttt{rec}\ f\ x\ \texttt{=>}\ e\ \underset{\sim}{\triangleleft\!\triangleright}\ \texttt{fn}\ x\ \texttt{=>}\ e\{(\texttt{rec}\ f\ x\ \texttt{=>}\ e)/f\}\ \overline{R^{\mathsf{C}}}\ \texttt{fn}\ x\ \texttt{=>}\ e\{v/f\} \lesssim v,$$

and we deduce $R \subseteq \langle \underset{\sim}{\triangleleft\!\triangleright}^{\circ}\ R^{\mathsf{C}}\ \lesssim^{\circ} \rangle$; hence $R \subseteq \lesssim$, by Proposition 10, and $\texttt{rec}\ f\ x\ \texttt{=>}\ e \lesssim v$, as required.

In analogy with Proposition 9 we would like to have a stronger rule:

$$\frac{R \subseteq \langle \underset{\sim}{\triangleright}^{\circ}\ (R^{\circ})^{\mathsf{C}}\ \lesssim^{\circ} \rangle}{R \subseteq \lesssim} \tag{7.1}$$

But we do not know if this holds. It would entail (5.14) which we left as an open problem. In Appendix A we prove a weaker version:

Lemma 7 $\dfrac{R\{\overline{R^{\mathsf{C}}}\} \cap Rel \subseteq \langle \gtrsim^{\circ}\; R^{\mathsf{C}} \lesssim^{\circ} \rangle}{R \subseteq \lesssim^{\circ}}$

It extends the open rule for simulation up to context (5.15) which we discussed in Section 5.1 as an approximation to (5.14).

When R is closed, Lemma 7 reduces to Proposition 10 above. But the utility of the lemma goes beyond that of Proposition 10 as we will now demonstrate by proving a syntactic minimal invariance property and by deriving equational rules for reasoning about open expressions.

7.1 Syntactic minimal invariance

As a non-trivial example, we consider a syntactic version of the domain-theoretic minimal invariance property for our language (Pitts 1994a).

Let π be the recursive function

$$\begin{aligned}\pi \stackrel{\text{def}}{=} \texttt{rec}\ f\ x \texttt{ => case}\ x\ \texttt{of nil => nil}&\\ \mid x_1 \texttt{ :: } x_2 \texttt{ => } f\ x_1 \texttt{ :: } f\ x_2&\\ \mid g \texttt{ => fn}\ y \texttt{ => } f(g(f\ y)).&\end{aligned}$$

(This would not be well-typed in ML where one would define a corresponding type-indexed family of functions instead.)

Minimal invariance says that π is the identity function, $\pi \sim \texttt{I}$. We can prove this by means of Lemma 7.

Proposition 11 (Syntactic minimal invariance) $\pi \sim \texttt{I}$.

Proof $\pi \lesssim \texttt{I}$ follows by recursion induction from

$$x \vdash \begin{array}{l}\texttt{case}\ x\ \texttt{of nil => nil}\\ \qquad \mid x_1 \texttt{ :: } x_2 \texttt{ => I}\ x_1 \texttt{ :: I}\ x_2\\ \qquad \mid g \texttt{ => fn}\ y \texttt{ => I}(g(\texttt{I}\ y))\end{array} \sim^{\circ} \quad x,$$

which is easily verified by case analysis on the value of x.

We invoke Lemma 7 to prove $\texttt{I} \lesssim \pi$. Let $R \subseteq Rel^{\circ}$ be given by

$$y \vdash y\ R\ \pi\, y,$$

and $S = \gtrsim^{\circ}\ R^{\mathsf{C}} \lesssim^{\circ}$. We prove $R\{\overline{R^{\mathsf{C}}}\} \cap Rel \subseteq \langle S \rangle$, i.e.,

$$\vdash v\ \overline{R^{\mathsf{C}}}\ v' \quad \text{implies} \quad \pi\ v' \Downarrow v'' \quad \text{with} \quad \vdash v\ \overline{S}\ v'',$$

by induction on the derivation of $\vdash v\ \overline{R^{\mathsf{C}}}\ v'$.

(Match nil) $v = \texttt{nil}$, $v' = \texttt{nil}$. Then $\pi\ v' \Downarrow \texttt{nil}$. Clearly $\vdash v\ \overline{S}\ \texttt{nil}$.

(Match cons) $v = v_1 \texttt{ :: } v_2$, $v' = v'_1 \texttt{ :: } v'_2$, and $\vdash v_i \ \overline{R^{\mathsf{C}}}\ v'_i$. By induction hypothesis $\pi\ v'_i \Downarrow v''_i$ with $\vdash v_i \ \overline{S}\ v''_i$. So $\pi\ v' \Downarrow v'' = v''_1 \texttt{ :: } v''_2$ and $\vdash v \ \overline{S}\ v''$.

(Match fn) $v = \texttt{fn}\ y \texttt{ => } e$, $v' = \texttt{fn}\ y \texttt{ => } e'$, and $y \vdash e \ R^{\mathsf{C}}\ e'$. Then $\pi\ v' \Downarrow v'' = \texttt{fn}\ y \texttt{ => } \pi(v'(\pi\ y))$ and $\vdash v \ \overline{S}\ v''$ because

$$
\begin{array}{rll}
y \vdash e & \mathrel{\underset{\sim}{\lhd\!\!\rhd}}^{\circ} & \texttt{let val}\ y = y\ \texttt{in let val}\ y = e\ \texttt{in}\ y\ \texttt{end end} \\
 & R^{\mathsf{C}} & \texttt{let val}\ y = \pi\ y\ \texttt{in let val}\ y = e'\ \texttt{in}\ \pi\ y\ \texttt{end end} \\
 & \sim^{\circ} & \pi(v'(\pi\ y)),
\end{array}
$$

and $\mathrel{\underset{\sim}{\lhd\!\!\rhd}}^{\circ}\ R^{\mathsf{C}} \sim^{\circ}\ \subseteq S$.

We conclude $R \subseteq\ \lesssim^{\circ}$ by Lemma 7. Therefore $v \lesssim \pi\, v$, for all closed values v, and $\mathtt{I} \lesssim \pi$ holds by extensionality. □

Simulation up to $\mathrel{\underset{\sim}{\rhd}}$ and context and $\lesssim$, Lemma 7, substantially simplifies the proof. Mason, Smith, and Talcott (1996) give a direct operational proof of this result. It is also possible to recast their proof in the relational proof style used throughout this paper.

The finite approximants of π (as defined in Section 4) are 'syntactic projections'. Syntactic minimal invariance and syntactic continuity entail that their least upper bound is the identity function and thus:

$$
p \lesssim q \quad \text{iff} \quad \forall n \geq 0.\ \pi_n\, p \lesssim q, \tag{7.2}
$$

where π_n is the n'th finite approximant of the recursive function π.

In Milner's construction of the fully abstract continuous model of PCF (Milner 1977) and in the operational model constructions for a call-by-value language like ours in Mason, Smith, and Talcott (1996), syntactic projections are used to address domain-theoretic notions of finite elements and ω-algebraicity syntactically.

Viewed as a proof rule, (7.2) is a sort of generalised Take Lemma (Bird and Wadler 1987) or higher-order structural induction principle; see Smith (1997). Pitts (1996b, 1994a) has also developed this idea and its domain-theoretic background and he has studied various applications.

7.2 Equational rules

From Lemma 7 we can derive an 'equational' version akin to Proposition 3.

Proposition 12 $\dfrac{R \subseteq\ \mathrel{\underset{\sim}{\rhd}}^{\circ}\ \overline{R^{\mathsf{C}}} \lesssim^{\circ}}{R \subseteq\ \lesssim^{\circ}}$

Proof If $R \subseteq\ \mathrel{\underset{\sim}{\rhd}}^{\circ}\ \overline{R^{\mathsf{C}}} \lesssim^{\circ}$ then

$$
\begin{array}{rcl}
R\{\overline{R^{\mathsf{C}}}\} & \subseteq & (\mathrel{\underset{\sim}{\rhd}}^{\circ}\ \overline{R^{\mathsf{C}}} \lesssim^{\circ})\{\overline{R^{\mathsf{C}}}\} \\
 & \subseteq & (\mathrel{\underset{\sim}{\rhd}}^{\circ}\{\overline{Id^{\circ}}\})(\overline{R^{\mathsf{C}}}\{\overline{R^{\mathsf{C}}}\})(\lesssim^{\circ}\{\overline{Id^{\circ}}\}) \\
 & \subseteq & \mathrel{\underset{\sim}{\rhd}}^{\circ}\ (\overline{R^{\mathsf{C}}}\{\overline{R^{\mathsf{C}}}\}) \lesssim^{\circ},
\end{array} \tag{7.3}
$$

because $\underset{\sim}{\trianglerighteq}^\circ$ and $\lesssim^\circ$ are closed under substitutions.

Moreover, we can show $\overline{R^{\mathsf{C}}}\{\overline{R^{\mathsf{C}}}\} \subseteq \underset{\sim}{\lhd\!\rhd}^\circ\ \overline{R^{\mathsf{C}}}\ \underset{\sim}{\lhd\!\rhd}^\circ$, i.e.,

$$\vec{x} \vdash u\ \overline{R^{\mathsf{C}}}\ u' \quad \text{and} \quad \vec{y} \vdash \vec{v}\ \overline{R^{\mathsf{C}}}\ \vec{v}' \quad \text{imply} \quad \vec{y} \vdash u\{\vec{v}/\vec{x}\}\ \underset{\sim}{\lhd\!\rhd}^\circ\ \overline{R^{\mathsf{C}}}\ \underset{\sim}{\lhd\!\rhd}^\circ\ u'\{\vec{v}'/\vec{x}\},$$

by induction on the derivation of $\vec{x} \vdash u\ \overline{R^{\mathsf{C}}}\ u'$. In the (Match fn) case the substitutions of u and u' into the function bodies can be replaced by let bindings up to cost equivalence.

Therefore $\underset{\sim}{\trianglerighteq}^\circ\ (\overline{R^{\mathsf{C}}}\{\overline{R^{\mathsf{C}}}\})\ \lesssim^\circ \quad \subseteq \quad \underset{\sim}{\trianglerighteq}^\circ\ \overline{R^{\mathsf{C}}}\ \lesssim^\circ$ and

$$\begin{array}{rcll}
R\{\overline{R^{\mathsf{C}}}\} \cap Rel & \subseteq & \underset{\sim}{\trianglerighteq}\ \overline{R^{\mathsf{C}}}\ \lesssim & \text{from (7.3)} \\
& \subseteq & \underset{\sim}{\trianglerighteq}\ \langle R^{\mathsf{C}} \rangle \lesssim & \text{by definition of } \langle _ \rangle \\
& \subseteq & \langle \underset{\sim}{\trianglerighteq}^\circ \rangle \langle R^{\mathsf{C}} \rangle \langle \lesssim^\circ \rangle & \underset{\sim}{\trianglerighteq} \text{ and } \lesssim \text{ are simulations} \\
& \subseteq & \langle \underset{\sim}{\trianglerighteq}^\circ\ R^{\mathsf{C}} \lesssim^\circ \rangle & \text{by (5.8),}
\end{array}$$

and we conclude $R \subseteq \underset{\sim}{\trianglerighteq}^\circ$, by Lemma 7. □

Proposition 12 is a useful proof rule in itself—for instance, the proof of recursion induction using Proposition 10 above is more directly an instance of the equational proof rule of Proposition 12. Furthermore, from it we can derive a version of a proof rule by Sands (1997b), called 'bisimulation up to context and improvement': let relation $\rhd \subseteq Rel$ be given by

$$p \rhd q \quad \text{iff} \quad p \underset{\sim}{\trianglerighteq} \mathtt{I}\ q.$$

Proposition 13 $\dfrac{R \subseteq \rhd^\circ\ R^{\mathsf{C}} \lesssim^\circ}{R \subseteq \lesssim^\circ}$

Proof Suppose $R \subseteq \rhd^\circ\ R^{\mathsf{C}} \lesssim^\circ$. We construct $S \subseteq Rel^\circ$ by

$$\vec{x} \vdash (\mathtt{fn}\ z \mathtt{\ =>\ } e) \quad S \quad (\mathtt{fn}\ z \mathtt{\ =>\ } e'),$$

whenever $z \notin \vec{x}$ and there exist d and d' such that

$$\vec{x} \vdash d\ R\ d' \quad \text{and} \quad \vec{x} \vdash d \rhd^\circ e\ R^{\mathsf{C}}\ e' \lesssim^\circ d'.$$

Observe that

$$\vec{x} \vdash d\ \underset{\sim}{\trianglerighteq}^\circ\ (\mathtt{fn}\ z \mathtt{\ =>\ } e)\,\mathtt{nil} \quad \text{and} \quad \vec{x} \vdash (\mathtt{fn}\ z \mathtt{\ =>\ } e')\,\mathtt{nil} \lesssim^\circ d'.$$

Therefore $R \subseteq \underset{\sim}{\trianglerighteq}^\circ\ S^{\mathsf{C}} \lesssim^\circ$. We have $S \subseteq \underset{\sim}{\trianglerighteq}^\circ\ \overline{S^{\mathsf{C}}} \lesssim^\circ$ because

$$\vec{x} \vdash (\mathtt{fn}\ z \mathtt{\ =>\ } e) \quad \overline{R^{\mathsf{C}}} \quad (\mathtt{fn}\ z \mathtt{\ =>\ } e'),$$

and

$$\overline{R^{\mathsf{C}}} \subseteq \overline{(\underset{\sim}{\trianglerighteq}^\circ\ S^{\mathsf{C}} \lesssim^\circ)^{\mathsf{C}}} \subseteq \overline{(\underset{\sim}{\trianglerighteq}^\circ)^{\mathsf{C}}}\ \overline{(S^{\mathsf{C}})^{\mathsf{C}}}\ \overline{(\lesssim^\circ)^{\mathsf{C}}} \subseteq \underset{\sim}{\trianglerighteq}^\circ\ \overline{S^{\mathsf{C}}} \lesssim^\circ.$$

Hence $S \subseteq \lesssim^\circ$, by Proposition 12, and $R \subseteq \underset{\sim}{\trianglerighteq}^\circ\ S^{\mathsf{C}} \lesssim^\circ$ implies $R \subseteq \lesssim^\circ$, because $\lesssim^\circ$ is a precongruence and contains $\underset{\sim}{\trianglerighteq}^\circ$. □

Sands (1997b) has demonstrated how (versions of) this rule allows simple calculational proofs of many functional program equivalences from the literature. It is particularly useful for call-by-value languages with inductively defined data types for which conventional applicative simulation is of little use.

For illustration, we solve Exercise 10.20 from Winskel (1993). Let

$$\begin{array}{lll}
f & \stackrel{\text{def}}{=} & \texttt{rec}\, f\ x \texttt{ => fn } y \texttt{ => case } x \texttt{ of nil => } y \\
 & & \qquad\qquad\qquad\qquad\qquad\quad \mid x_1 \texttt{ :: } x_2 \texttt{ => } f(\texttt{append}(r x_1 y) x_2)(s x_1 y) \\
 & & \qquad\qquad\qquad\qquad\qquad\quad \mid h \texttt{ => } \Omega, \\
g & \stackrel{\text{def}}{=} & \texttt{rec}\, g\ x \texttt{ => fn } y \texttt{ => case } x \texttt{ of nil => } y \\
 & & \qquad\qquad\qquad\qquad\qquad\quad \mid x_1 \texttt{ :: } x_2 \texttt{ => } g x_2 (g(r x_1 y)(s x_1 y)) \\
 & & \qquad\qquad\qquad\qquad\qquad\quad \mid h \texttt{ => } \Omega,
\end{array}$$

where $r, s \in \mathit{Val}_{\emptyset}$ (presumably they are functions but we need not require that) and `append` is the list concatenation function,

$$\begin{array}{lll}
\texttt{append} & \stackrel{\text{def}}{=} & \texttt{rec}\, a\ x \texttt{ => fn } y \texttt{ => case } x \texttt{ of nil => } y \\
 & & \qquad\qquad\qquad\qquad\qquad\quad \mid x_1 \texttt{ :: } x_2 \texttt{ => } x_1 \texttt{ :: } a\, x_2\, y \\
 & & \qquad\qquad\qquad\qquad\qquad\quad \mid g \texttt{ => } \Omega.
\end{array}$$

We will now prove that f and g are bisimilar by means of Sands' proof rule. As a first attempt, let relation R be given by

$$\vec{x} \vdash (f\, u_0\, v)\ R\ (g\, u_0\, v),$$

whenever $u_0, v \in \mathit{Val}_{\vec{x}}$. By extensionality, $f \sim g$ if $R \subseteq \sim^{\circ}$.

By means of the equational laws for cost equivalence, we calculate

$$\begin{array}{lll}
\vec{x} \vdash f\, u_0\, v & \underset{\sim}{\triangleleft\!\triangleright}^{\circ} & \texttt{I}^2\, \texttt{case } u_0 \texttt{ of nil => } v \\
 & & \qquad\qquad \mid x_1 \texttt{ :: } x_2 \texttt{ => let val } x = r\, x_1\, v \\
 & & \qquad\qquad\qquad\qquad\qquad \texttt{in let val } y = s\, x_1\, v \\
 & & \qquad\qquad\qquad\qquad\qquad\quad \texttt{in } f(\texttt{append}\, x\, x_2)\, y \\
 & & \qquad\qquad\qquad\qquad\qquad\quad \texttt{end} \\
 & & \qquad\qquad\qquad\qquad\qquad \texttt{end} \\
 & & \qquad\qquad \mid h \texttt{ => } \Omega, \\
\vec{x} \vdash g\, u_0\, v & \underset{\sim}{\triangleleft\!\triangleright}^{\circ} & \texttt{I}^2\, \texttt{case } u_0 \texttt{ of nil => } v \\
 & & \qquad\qquad \mid x_1 \texttt{ :: } x_2 \texttt{ => let val } x = r\, x_1\, v \\
 & & \qquad\qquad\qquad\qquad\qquad \texttt{in let val } y = s\, x_1\, v \\
 & & \qquad\qquad\qquad\qquad\qquad\quad \texttt{in } g\, x_2 (g\, x\, y) \\
 & & \qquad\qquad\qquad\qquad\qquad\quad \texttt{end} \\
 & & \qquad\qquad\qquad\qquad\qquad \texttt{end} \\
 & & \qquad\qquad \mid h \texttt{ => } \Omega.
\end{array}$$

The resulting expressions are identical except for the subterms $f(\texttt{append}\, x\, x_2)\, y$ and $g\, x_2(g\, x\, y)$. We need to extend R to also relate these. Let $@(u_n, \ldots, u_0)$ abbreviate $\texttt{append}\, u_n(\texttt{append}\, u_{n-1}(\ldots(\texttt{append}\, u_1\, u_0)\ldots))$. Now, let R be given by

$$\vec{x} \vdash (f\ @(u_n, \ldots, u_0)\, v)\ R\ (g\, u_0(\ldots(g\, u_n\, v)\ldots)),$$

for all $n \geq 0$ and $u_0,\ldots, u_n, v \in \mathit{Val}_{\vec{x}}$. From the calculations above we see that

$$\vec{x} \vdash (f \,@(u_0)\, v) = (f\, u_0\, v) \rhd^\circ R^{\mathsf{C}} \sim^\circ (g\, u_0\, v),$$

because $\vec{x}x_1x_2xy \vdash (f(\texttt{append}\, x\, x_2)\, y) = (f\, @(x, x_2)\, y) \;\; R \;\; (g\, x_2(g\, x\, y))$. If $n \geq 1$ we calculate

$$
\begin{array}{ll}
\multicolumn{2}{l}{\vec{x} \vdash f\,@(u_n, u_{n-1},\ldots, u_0)\, v} \\
\underset{\sim}{\lhd\!\rhd}^\circ & \texttt{I}^2\, \texttt{case}\ u_n\ \texttt{of nil => } f\,@(u_{n-1},\ldots, u_0)\, v \\
& \qquad \texttt{|}\ x_1\ \texttt{::}\ x_2\ \texttt{=> let val}\ x = r\, x_1\, v \\
& \qquad\qquad \texttt{in let val}\ y = s\, x_1\, v \\
& \qquad\qquad\quad \texttt{in}\ \texttt{I}^2\, f\,@(x, x_2, u_{n-1},\ldots, u_0)\, y \\
& \qquad\qquad\quad \texttt{end} \\
& \qquad\qquad \texttt{end} \\
& \qquad \texttt{|}\ h\ \texttt{=>}\ \Omega, \\
\underset{\sim}{\rhd}^\circ & \texttt{I}^2\, \texttt{case}\ u_n\ \texttt{of nil => } f\,@(u_{n-1},\ldots, u_0)\, v \\
& \qquad \texttt{|}\ x_1\ \texttt{::}\ x_2\ \texttt{=> let val}\ x = r\, x_1\, v \\
& \qquad\qquad \texttt{in let val}\ y = s\, x_1\, v \\
& \qquad\qquad\quad \texttt{in}\ f\,@(x, x_2, u_{n-1},\ldots, u_0)\, y \\
& \qquad\qquad\quad \texttt{end} \\
& \qquad\qquad \texttt{end} \\
& \qquad \texttt{|}\ h\ \texttt{=>}\ \Omega, \\
\multicolumn{2}{l}{\vec{x} \vdash g\, u_0(\ldots(g\, u_{n-1}(g\, u_n\, v))\ldots)} \\
\underset{\sim}{\lhd\!\rhd}^\circ & \texttt{I}^2\, \texttt{case}\ u_0\ \texttt{of nil => } g\, u_0(\ldots(g\, u_{n-1}\, v)\ldots) \\
& \qquad \texttt{|}\ x_1\ \texttt{::}\ x_2\ \texttt{=> let val}\ x = r\, x_1\, v \\
& \qquad\qquad \texttt{in let val}\ y = s\, x_1\, v \\
& \qquad\qquad\quad \texttt{in}\ g\, u_0(\ldots(g\, u_{n-1}(g\, x_2(g\, x\, y)))\ldots) \\
& \qquad\qquad\quad \texttt{end} \\
& \qquad\qquad \texttt{end} \\
& \qquad \texttt{|}\ h\ \texttt{=>}\ \Omega.
\end{array}
$$

Since $\vec{x}x_1x_2xy \vdash (f\,@(x, x_2, u_{n-1},\ldots, u_0)\, y) \;\; R \;\; (g\, u_0(\ldots(g\, u_{n-1}(g\, x_2(g\, x\, y)))\ldots))$ and $\vec{x} \vdash (f\,@(u_{n-1},\ldots, u_0)\, v) \;\; R \;\; (g\, u_0(\ldots(g\, u_{n-1}\, v)\ldots))$, we get that

$$\vec{x} \vdash (f\,@(u_n, u_{n-1},\ldots, u_0)\, v) \rhd^\circ R^{\mathsf{C}} \sim^\circ (g\, u_0(\ldots(g\, u_{n-1}(g\, u_n\, v))\ldots)).$$

Hence $R \subseteq \rhd^\circ R^{\mathsf{C}} \sim^\circ$ and thus $R \subseteq \lesssim^\circ$ by Proposition 13. From the calculations above it is easy to obtain $R^{\mathrm{op}} \subseteq \rhd^\circ (R^{\mathrm{op}})^{\mathsf{C}} \sim^\circ$ too, and hence $R^{\mathrm{op}} \subseteq \lesssim^\circ$, again by Proposition 13. We conclude that $R \subseteq \sim^\circ$ and $f \sim g$.

The shortcoming of Lemma 7, compared to (7.1), is less apparent in the derived equational rules of Propositions 12 and 13. But note that they work for *open* relations, in contrast to the 'closed' equational rule for simulation up to similarity of Proposition 3. We do not know if stronger, closed versions hold:

$$(i)\ \frac{R \subseteq \underset{\sim}{\rhd}\, \overline{(R^\circ)^{\mathsf{C}}} \lesssim}{R \subseteq \lesssim} \qquad\qquad (ii)\ \frac{R \subseteq \rhd\, (R^\circ)^{\mathsf{C}} \lesssim}{R \subseteq \lesssim} \tag{7.4}$$

They are consequences of (7.1) because $\gtrsim \overline{(R^\circ)^{\mathsf{C}}} \lesssim \; \subseteq \langle \gtrsim^\circ (R^\circ)^{\mathsf{C}} \lesssim^\circ \rangle$ and (ii) follows from (i) as in the proof of Proposition 13. Propositions 12 and 13 are weaker than (7.4): sometimes reasoning about open terms does not suffice as it may be necessary to argue by cases on the values of the free variables. One such example is syntactic minimal invariance, Proposition 11. It would follow from (7.4)(i), by structural induction on closed values, but not from Proposition 12.

8 Conclusion

The 'relational' proof style of Howe (1996) and Pitts (1995) has been used throughout this paper. It is a rather low-level approach but is precise and tractable and applies to a wide range of problems involving term contexts and evaluation. Our proofs of the unfolding theorem and various simulation up to context results substantiate this. The algebra of relations in Section 3 and, in particular, context closure facilitate the construction of relations for this style of proofs. Our results are stated for an untyped ML fragment but should carry over to other typed and untyped higher-order languages.

Simulation up to context is a proof technique with a great practical potential for applicative bisimulation and improvement. This is witnessed by our proofs of recursion induction, the improvement theorem, syntactic minimal invariance, and Exercise 10.20 from Winskel (1993), as well as by the examples of Gordon (1995) and Sands (1997b). But an important problem is left open, namely the validity of (5.14) and (7.1),

$$\frac{R \subseteq \langle (R^\circ)^{\mathsf{C}} \rangle}{R \subseteq \lesssim} \qquad\qquad \frac{R \subseteq \langle \gtrsim^\circ (R^\circ)^{\mathsf{C}} \lesssim^\circ \rangle}{R \subseteq \lesssim}$$

The significance of the gap between these and the weaker rule of Lemma 7,

$$\frac{R\{\overline{R^{\mathsf{C}}}\} \cap Rel \subseteq \langle \gtrsim^\circ R^{\mathsf{C}} \lesssim^\circ \rangle}{R \subseteq \lesssim^\circ}$$

is unclear. In Section 7 we demonstrated how Lemma 7 allows us to prove a range of non-trivial results.

Acknowledgements After some time of fruitless investigations into applicative bisimulation up to context, this work got under way when Andrew Pitts showed me his notes on the subject (Pitts 1995). I am grateful for his encouragement and valuable guidance in this project. I wish to thank David Sands for discussions; his work has been influential in every aspect of this research. Finally, I am indebted to Andrew Gordon, Peter Ørbæk and an anonymous referee for detailed comments and helpful suggestions. I am supported by a grant from the Danish Natural Science Research Council.

A Proofs

This appendix contains the rather delicate proofs of Lemmas 6 and 7. The first of these uses the following lemma.

Lemma 8 *Compatibility is preserved by transitive closure.*

Proof First observe that compatibility is preserved by relation composition: if R and S are compatible, so is their composition $R\,S$,

$$\widehat{R\,S} = \widehat{R}\,\widehat{S} \subseteq R\,S.$$

Next, suppose R is compatible. If $\vec{x} \vdash e \ \widehat{R^+} \ e'$, each immediate subterm e_i of e is related to a corresponding subterm e'_i of e', $\vec{x}\vec{y_i} \vdash e_i \ R^+ \ e'_i$, for some $\vec{y_i}$. This means that there exists $m_i \geq 1$ such that $\vec{x}\vec{y_i} \vdash e_i \ R^{m_i} \ e'_i$, where R^{m_i} is the m_i-fold composition of R with itself. Let m be the greatest of these m_i, for all pairs of subterms. Since R is compatible it is also reflexive. Hence $\vec{x}\vec{y_i} \vdash e_i \ R^{m-m_i} \ e_i$ and then $\vec{x}\vec{y_i} \vdash e_i \ R^m \ e'_i$, for all corresponding subterms e_i and e'_i. Hence $\vec{x} \vdash e \ \widehat{R^m} \ e'$, by definition of compatible refinement, and then $\vec{x} \vdash e \ R^m \ e'$ because compatibility is preserved by relation composition. So $\vec{x} \vdash e\ R^+\ e'$ and we conclude that R^+ is compatible. □

Proof of Lemma 6 $\dfrac{R \subseteq \langle S^+ \rangle_I}{S^+ \subseteq \gtrsim^\circ}$, where $S \stackrel{\text{def}}{=} (R^\circ)^{\mathsf{C}}$.

Proof Assume $R \subseteq \langle S^+ \rangle_I$. We are going to prove that S^+ is an open improvement simulation,

$$S^+ \subseteq \langle S^+ \rangle_I^\circ.$$

Then $S^+ \subseteq \gtrsim^\circ$, by the improvement simulation rule (6.2).

First we need some properties of S^+.

By definition of context closure, S is compatible. Compatibility is preserved by transitive closure, Lemma 8, so S^+ is also compatible.

Open extension, R°, is closed under substitutions and so is S because closure under substitutions is preserved by context closure. It is also preserved by relation composition and, consequently, by transitive closure. Therefore S^+ is closed under substitutions.

By Lemma 1, S^+ is substitutive because it is compatible, transitive and closed under substitutions.

We proceed to prove $S^+ \subseteq \langle S^+ \rangle_I^\circ$. Since S^+ is closed under substitutions, it suffices to show $S^+ \cap \mathit{Rel} \subseteq \langle S^+ \rangle_I$. This is equivalent to asserting that predicate $\mathcal{P}(N)$, defined by

$$\mathcal{P}(N) \stackrel{\text{def}}{\Leftrightarrow} \begin{array}{l} \forall p, p', v.\ \vdash p\ S^+\ p' \ \ \&\ \ p \Downarrow_N v \ \Rightarrow \\ \qquad \exists v'.\ p' \Downarrow_{\leq N} v' \ \ \&\ \ \vdash v\ \overline{S^+}\ v', \end{array}$$

holds for all N. The proof is by a series of nested inductions on N, on the derivation of $\vdash p\ S^+\ p'$, and on the derivation of $p \Downarrow_N v$.

The outer induction hypothesis is

(I.H.1) $\mathcal{P}(N)$ *for all* $N < N_0$.

Then we must show $\mathcal{P}(N_0)$. This follows if

$$\forall p, p', v.\ \vdash p\ S\ p'\ \ \&\ \ p \Downarrow_{\leq N_0} v\ \Rightarrow\ \exists v'.\ p' \Downarrow_{\leq N_0} v'\ \ \&\ \ \vdash v\ \overline{S^+}\ v', \tag{A.1}$$

because, supposing $\vdash p\ S^+\ p'$,

$$\vdash p = p_0\ S\ p_1\ S \cdots S\ p_m = p',$$

and $p \Downarrow_{N_0} v$, by repeatedly applying (A.1) to $\vdash p_i\ S\ p_{i+1}$, we get $p' \Downarrow_{\leq N_0} v'$ with

$$\vdash v = v_0\ \overline{S^+} \cdots \overline{S^+}\ v_m = v',$$

and we can conclude $\vdash v\ \overline{S^+}\ v'$ because $\overline{S^+}$ is transitive,

$$\overline{S^+} \cdots \overline{S^+} = \overline{S^+ \cdots S^+} \subseteq \overline{S^+}.$$

We strengthen (A.1) slightly and prove that predicate $\mathcal{Q}(p, M, v)$,

$$\mathcal{Q}(p, M, v)\ \ \stackrel{\text{def}}{\Leftrightarrow}\ \ \begin{array}{l} p \Downarrow_M v\ \ \&\ \ M \leq N_0\ \Rightarrow \\ \qquad \forall p'.\ \vdash p\ S\{\overline{S^+}\}\ p'\ \Rightarrow \\ \qquad\qquad \exists v'.\ p' \Downarrow_{\leq M} v'\ \ \&\ \ \vdash v\ \overline{S^+}\ v', \end{array}$$

holds for all p, M, v. This entails (A.1) as $\vdash p\ S\ p'$ clearly implies $\vdash p\ S\{\overline{S^+}\}\ p'$ with empty substitution of $\overline{S^+}$. Observe also that $S\{\overline{S^+}\} \subseteq S^+$ because $\overline{S^+} \subseteq S^+$, by compatibility, and $S\{S^+\} \subseteq S^+$, since $S \subseteq S^+$ and S^+ is substitutive. In fact, $\mathcal{Q}(p, M, v)$ follows from (I.H.1) whenever $M < N_0$.

We prove $\mathcal{Q}(p, M, v)$, for all p, M, v, by induction on the derivation of $p \Downarrow_M v$. For any derivation $p_0 \Downarrow_{M_0} v_0$, the induction hypothesis is

(I.H.2) $\mathcal{Q}(p, M, v)$ *for all premises* $p \Downarrow_M v$ *in the derivation of* $p_0 \Downarrow_{M_0} v_0$,

and we must show $\mathcal{Q}(p_0, M_0, v_0)$. We assume $p_0 \Downarrow_{M_0} v_0$ and $M_0 \leq N_0$, where $p_0 = e\{\vec{u}/\vec{x}\}$, $\vec{x} \vdash e\ S\ e'$ and $\vdash \vec{u}\ \overline{S^+}\ \vec{u}'$ such that $\vdash e\{\vec{u}/\vec{x}\}\ S\{\overline{S^+}\}\ e'\{\vec{u}'/\vec{x}\}$. We will show $e'\{\vec{u}'/\vec{x}\} \Downarrow_{\leq M_0} v_0'$, for some v_0' such that $\vdash v_0\ \overline{S^+}\ v_0'$.

The strategy is to exploit the assumption $\vec{x} \vdash e\ S\ e'$ to build the derivation $e'\{\vec{u}'/\vec{x}\} \Downarrow_{\leq M_0} v_0'$. The substitutions of $\vec{u}$ and $\vec{u}'$ are separated out in the induction on the derivation of $e\{\vec{u}/\vec{x}\} \Downarrow_{M_0} v_0$. If it is derived by means of the (Eval apply) rule, we need to perform these substitutions and we end up with terms related by S^+ rather than $S\{\overline{S^+}\}$. Then we invoke the stronger induction hypothesis (I.H.1). It applies because the premises of the (Eval apply) rule will all have cost-indexes smaller than M_0 and N_0.

Recall $S = (R^\circ)^{\mathsf{C}}$ and consider the derivation of $\vec{x} \vdash e\ S\ e'$. There are two cases.

(Ctx R) Suppose $\vec{y} \vdash e\ R^\circ\ e'$ with $\vec{y} \subseteq \vec{x}$. Note that $\vec{x} \vdash e\ \widehat{S}\ e$ because S and $\widehat{S}$ are reflexive. Therefore $\vdash e\{\vec{u}/\vec{x}\}\ \widehat{S}\{\overline{S^+}\}\ e\{\vec{u}'/\vec{x}\}$. From the (Ctx Comp) case below we get $e\{\vec{u}'/\vec{x}\} \Downarrow_{\leq M_0} v_0''$ with $\vdash v_0\ \overline{S^+}\ v_0''$. Moreover, $\vdash e\{\vec{u}'/\vec{x}\}\ R\ e'\{\vec{u}'/\vec{x}\}$. The assumption $R \subseteq \langle S^+\rangle_I$ implies $e'\{\vec{u}'/\vec{x}\} \Downarrow_{\leq M_0} v_0'$ with $\vdash v_0''\ \overline{S^+}\ v_0'$. Since $\overline{S^+}$ is transitive, we obtain $\vdash v_0\ \overline{S^+}\ v_0'$, as required.

(Ctx Comp) Suppose $\vec{x} \vdash e\ \widehat{S}\ e'$. If this is derived by the (Comp x) rule, then $e = e' = x_i$, for some $x_i \in \vec{x}$, and the result is immediate because $e\{\vec{u}/\vec{x}\} \Downarrow_0 u_i$, $e'\{\vec{u}'/\vec{x}\} \Downarrow_0 u_i'$ and, by assumption, $\vdash u_i\ \overline{S^+}\ u_i'$. Otherwise e and e' are not variables, and we proceed by analysis of the derivation of $e\{\vec{u}/\vec{x}\} \Downarrow_{M_0} v_0$.

Case (Eval fn) $e\{\vec{u}/\vec{x}\}$ is a function, $e\{\vec{u}/\vec{x}\} = v_0$, and $M_0 = 0$. Since e is not a variable, it is itself a function, $e = \texttt{fn}\ y\ \texttt{=>}\ d$ for some $d \in \mathit{Exp}_{\vec{x}y}$. Then $\vec{x} \vdash e\ \widehat{S}\ e'$ must be derived by (Comp fn) so that $e' = \texttt{fn}\ y\ \texttt{=>}\ d'$ where $\vec{x}y \vdash d\ S\ d'$. Hence $e'\{\vec{u}'/\vec{x}\} \Downarrow_0 e'\{\vec{u}'/\vec{x}\}$ and $\vdash e\{\vec{u}/\vec{x}\}\ \overline{S^+}\ e'\{\vec{u}'/\vec{x}\}$, by (Eval fn) and (Match fn) because $y \vdash d\{\vec{u}/\vec{x}\}\ S\{\overline{S^+}\}\ d'\{\vec{u}'/\vec{x}\}$ and $S\{\overline{S^+}\} \subseteq S^+$.

Case (Eval nil) By reasoning similar to the previous case, we see e and e' are both $\texttt{nil}$. Hence $e'\{\vec{u}'/\vec{x}\} \Downarrow_0 e'\{\vec{u}'/\vec{x}\}$ and $\vdash e\{\vec{u}/\vec{x}\}\ \overline{S^+}\ e'\{\vec{u}'/\vec{x}\}$, by (Eval nil) and (Match nil).

Case (Eval cons) Since e is not a variable, it must be of the form $e = e_1\ \texttt{::}\ e_2$, and $e_i\{\vec{u}/\vec{x}\} \Downarrow_{M_i} v_i$, for $i = 1, 2$, such that $M_0 = M_1 + M_2$ and $v_0 = v_1\ \texttt{::}\ v_2$. Then $\vec{x} \vdash e\ \widehat{S}\ e'$ implies $e' = e_1'\ \texttt{::}\ e_2'$ and $\vec{x} \vdash e_i\ S\ e_i'$. Observe that $M_i \leq N_0$ because $M_i \leq M_0$ and $M_0 \leq N_0$. Induction hypothesis (I.H.2) and $e_i\{\vec{u}/\vec{x}\} \Downarrow_{M_i} v_i$ imply $e_i'\{\vec{u}'/\vec{x}\} \Downarrow_{\leq M_i} v_i'$ with $\vdash v_i\ \overline{S^+}\ v_i'$. Hence $e'\{\vec{u}'/\vec{x}\} \Downarrow_{\leq M_0} v_0'$ with $\vdash v_0 = v_1\ \texttt{::}\ v_2\ \overline{S^+}\ v_1'\ \texttt{::}\ v_2'$, by (Eval Cons) and (Match Cons).

Case (Eval apply) $e = e_1 e_2$, $e_1\{\vec{u}/\vec{x}\} \Downarrow_{M_1} v_1 = \texttt{fn}\ y\ \texttt{=>}\ d$, $e_2\{\vec{u}/\vec{x}\} \Downarrow_{M_2} v_2$, $d\{v_2/y\} \Downarrow_{M_3} v_0$, and $M_0 = M_1 + M_2 + M_3 + 1$. Since $\vec{x} \vdash e\ \widehat{S}\ e'$, $e' = e_1' e_2'$ with $\vec{x} \vdash e_i\ S\ e_i'$. We observe that $M_i < N_0$ because $M_i < M$ and $M \leq N_0$, for $i = 1, 2, 3$. By induction hypothesis (I.H.2), $e_i\{\vec{u}/\vec{x}\} \Downarrow_{M_i} v_i$ implies $e_i'\{\vec{u}'/\vec{x}\} \Downarrow_{\leq M_i} v_i'$ with $\vdash v_i\ \overline{S^+}\ v_i'$, for $i = 1, 2$. So $v_1' = \texttt{fn}\ y\ \texttt{=>}\ d'$ where $y \vdash d\ S^+\ d'$. Hence $\vdash d\{v_2/y\}\ S^+\ d'\{v_2'/y\}$, because S^+ is substitutive. Since $M_3 < N_0$, induction hypothesis (I.H.1) and $d\{v_2/y\} \Downarrow_{M_3} v_0$ imply $d'\{v_2'/y\} \Downarrow_{\leq M_3} v_0'$ with $\vdash v_0\ \overline{S^+}\ v_0'$. By (Eval apply), we conclude $e'\{\vec{u}'/\vec{x}\} \Downarrow_{\leq M_0} v_0'$.

Case (Eval let fun) $e = \texttt{let fun}\ f\ y = e_1\ \texttt{in}\ e_2\ \texttt{end}$, $e_2\{\vec{u}w/\vec{x}f\} \Downarrow_{M_0} v_0$, where $w = \texttt{rec}\ f\ y\ \texttt{=>}\ e_1$. Since $\vec{x} \vdash e\ \widehat{S}\ e'$, $e' = \texttt{let fun}\ f\ y\texttt{=}e_1'\ \texttt{in}\ e_2'\ \texttt{end}$ with $\vec{x}fy \vdash e_1\ S\ e_1'$ and $\vec{x}f \vdash e_2\ S\ e_2'$. Let $w' = \texttt{rec}\ f\ y\ \texttt{=>}\ e_1'$, then $\vec{x} \vdash w\ \widehat{S}\ w'$ because S is compatible. Consequently, $\vec{x} \vdash w\ \overline{S^+}\ w'$ and $\vdash e_2\{\vec{u}w/\vec{x}f\}\ S\{\overline{S^+}\}\ e_2'\{\vec{u}'w'/\vec{x}f\}$. By induction hypothesis (I.H.2), $e_2\{\vec{u}w/\vec{x}f\} \Downarrow_{M_0} v_0$ implies $e_2'\{\vec{u}'w'/\vec{x}f\} \Downarrow_{\leq M_0} v_0'$ with $\vdash v_0\ \overline{S^+}\ v_0'$. By (Eval let fun), we conclude $e'\{\vec{u}'/\vec{x}\} \Downarrow_{\leq M_0} v_0'$.

Case (Eval case) $e = \texttt{case}\ e_0\ \texttt{of nil => }\ e_1\ \texttt{|}\ y_1\ \texttt{::}\ y_2\ \texttt{=>}\ e_2\ \texttt{|}\ f\ \texttt{=>}\ e_3$, $e_0\{\vec{u}/\vec{x}\} \Downarrow_{M_1} v$, $q \Downarrow_{M_2} v_0$, and $M_0 = M_1 + M_2$, where

$$q = \begin{cases} e_1\{\vec{u}/\vec{x}\} & \text{if } v = \texttt{nil} \\ e_2\{\vec{u}v_1v_2/\vec{x}y_1y_2\} & \text{if } v = v_1\ \texttt{::}\ v_2 \\ e_3\{\vec{u}v/\vec{x}f\} & \text{if } v = \texttt{fn}\ y\ \texttt{=>}\ d. \end{cases}$$

Since $\vec{x} \vdash e\ \widehat{S}\ e'$, $e' = \texttt{case}\ e'_0\ \texttt{of nil =>}\ e'_1\ \texttt{|}\ y_1\ \texttt{::}\ y_2\ \texttt{=>}\ e'_2\ \texttt{|}\ f\ \texttt{=>}\ e'_3$, with $\vec{x} \vdash e_0\ S\ e'_0$, $\vec{x} \vdash e_1\ S\ e'_1$, $\vec{x}y_1y_2 \vdash e_2\ S\ e'_2$, and $\vec{x}f \vdash e_3\ S\ e'_3$. Observe that $M_i \leq N_0$ because $M_i \leq M_0$ and $M_0 \leq N_0$. Induction hypothesis (I.H.2) and $e_0\{\vec{u}/\vec{x}\} \Downarrow_{M_1} v$ imply $e'_0\{\vec{u}'/\vec{x}\} \Downarrow_{\leq M_1} v'$ with $\vdash v\ \overline{S^+}\ v'$. According to the derivation of $\vdash v\ \overline{S^+}\ v'$, let

$$q' = \begin{cases} e'_1\{\vec{u}'/\vec{x}\} & \text{if } v = \texttt{nil} = v' \\ e'_2\{\vec{u}'v'_1v'_2/\vec{x}y_1y_2\} & \text{if } v = v_1\ \texttt{::}\ v_2, v' = v'_1\ \texttt{::}\ v'_2, \vdash v_i\ \overline{S^+}\ v'_i \\ e'_3\{\vec{u}'v'/\vec{x}f\} & \text{if } \vdash v = \texttt{fn}\ y\ \texttt{=>}\ d\ \overline{S^+}\ \texttt{fn}\ y\ \texttt{=>}\ d' = v'. \end{cases}$$

By inspection of the definitions of q and q', we see that $\vdash q\ S\{\overline{S^+}\}\ q'$. Induction hypothesis (I.H.2) and $q \Downarrow_{M_2} v_0$ imply $q' \Downarrow_{\leq M_2} v'_0$ with $\vdash v_0\ \overline{S^+}\ v'_0$. By (Eval case), we conclude $e'\{\vec{u}'/\vec{x}\} \Downarrow_{\leq M_0} v'_0$.

Case (Eval let val) Similar to the previous case.

All cases considered, we conclude $\mathcal{Q}(p_0, M_0, v_0)$, as required. This completes the induction step for $\mathcal{Q}$ and we have $\mathcal{Q}(p, M, v)$ for all p, M, v.

Therefore (A.1) and $\mathcal{P}(N_0)$ hold. This completes the induction step for $\mathcal{P}$, so $\mathcal{P}(N)$ holds for all N.

Then $S^+ \cap \mathit{Rel} \subseteq \langle S^+ \rangle_I$ and S^+ is an open improvement simulation, $S^+ \subseteq \langle S^+ \rangle_I^\circ$. The conclusion, $S^+ \subseteq \gtrsim^\circ$, follows co-inductively by the improvement simulation rule (6.2). □

Lemma 6 is invalid for similarity (i.e., with $\langle \_ \rangle$ and $\lesssim^\circ$ in place of $\langle \_ \rangle_I$ and $\gtrsim^\circ$). The proof above uses the improvement aspect of $\langle \_ \rangle_I$ to assert that $p' \Downarrow v'$ computes no slower than $p \Downarrow v$ when $\vdash p\ S\ p'$. If there is no bound on the cost of $p' \Downarrow v'$, the transitivity argument, why (A.1) implies $\mathcal{P}(N_0)$, breaks down.

The proof of Lemma 6 makes use of the transitive closure of S in two ways.

(i) S^+ is substitutive. This is used in the (Eval fn) and (Eval apply) cases of the induction step.

(ii) In the (Ctx R) case transitivity is used to avoid substitution of values related by $\overline{S^+}$ into expressions related by R°.

Lemma 7 is essentially a weaker version of Lemma 6 for similarity. The proof is similar to that of Lemma 6 but solves (i) and (ii) without transitive closure.

(i) Substitutions can be replaced by let bindings up to cost equivalence; thereby the compatibility of R^{C} suffices and substitutivity is not necessary.

(ii) The requirement that the premise of Lemma 7 holds for $R\{\overline{R^{\mathsf{C}}}\}$ rather than just R circumvents the problem of substitution into expressions related by R.

Proof of Lemma 7 $\dfrac{R\{\overline{R^{\mathsf{C}}}\} \cap Rel \subseteq \langle \underset{\sim}{\triangleright}^\circ\ R^{\mathsf{C}} \lesssim^\circ \rangle}{R \subseteq \lesssim^\circ}$

Proof Assume $R\{\overline{R^{\mathsf{C}}} \cap Rel\} \subseteq \langle \underset{\sim}{\triangleright}^\circ\ R^{\mathsf{C}} \lesssim^\circ \rangle$. We are going to prove

$$R^{\mathsf{C}}\{Id\} \subseteq \langle S \rangle, \quad \text{where } S = \underset{\sim}{\triangleright}^\circ\ R^{\mathsf{C}} \lesssim^\circ. \tag{A.2}$$

Observe that $S \subseteq \lesssim^\circ R^{\mathsf{C}}\{Id\}^\circ \lesssim^\circ$, because $\underset{\sim}{\triangleright}^\circ \subseteq \lesssim^\circ$ and $R^{\mathsf{C}} \subseteq R^{\mathsf{C}}\{Id\}^\circ$, by (3.2). By simulation up to similarity, Proposition 2, we get that (A.2) implies $R^{\mathsf{C}}\{Id\} \subseteq \lesssim$. Hence $R^{\mathsf{C}} \subseteq \lesssim^\circ$, by (3.2), and the conclusion, $R \subseteq \lesssim^\circ$, follows because $R \subseteq R^{\mathsf{C}}$,

S is compatible because $\underset{\sim}{\triangleright}^\circ$, R^{C}, and $\lesssim^\circ$ are and compatibility is preserved by relation composition. Therefore S is reflexive and $\overline{S}$ is a reflexive relation on values. Hence $R^{\mathsf{C}}\{Id\} \subseteq R^{\mathsf{C}}\{\overline{S}\}$, and (A.2) holds if $R^{\mathsf{C}}\{\overline{S}\} \subseteq \langle S \rangle$. The latter is equivalent to the predicate $\mathcal{P}(N)$ holding for all N, where

$$\mathcal{P}(N) \overset{\text{def}}{\Leftrightarrow} \begin{array}{l} \forall p, p', v.\ \vdash p\ R^{\mathsf{C}}\{\overline{S}\}\ p' \ \&\ p \Downarrow_N v \ \Rightarrow \\ \qquad \exists v'.\ p' \Downarrow v' \ \&\ \vdash v\ \overline{S}\ v', \end{array}$$

We proceed by induction on N. The induction hypothesis is

(I.H.1) $\mathcal{P}(N)$ *for all* $N < N_0$.

Then we must show $\mathcal{P}(N_0)$. This follows if $\mathcal{Q}(p, M, v)$,

$$\mathcal{Q}(p, M, v) \overset{\text{def}}{\Leftrightarrow} \begin{array}{l} p \Downarrow_M v \ \&\ M \le N_0 \ \Rightarrow \\ \qquad \forall p'.\ \vdash p\ R^{\mathsf{C}}\{\overline{S}\}\ p' \ \Rightarrow \\ \qquad\qquad \exists v'.\ p' \Downarrow v' \ \&\ \vdash v\ \overline{S}\ v', \end{array}$$

holds for all p, M, v. We prove $\mathcal{Q}(p, M, v)$, for all p, M, v, by induction on the derivation of $p \Downarrow_M v$. For any derivation $p_0 \Downarrow_{M_0} v_0$, the induction hypothesis is

(I.H.2) $\mathcal{Q}(p, M, v)$ *for all premises* $p \Downarrow_M v$ *in the derivation of* $p_0 \Downarrow_{M_0} v_0$*,*

and we must show $\mathcal{Q}(p_0, M_0, v_0)$. We assume $p_0 \Downarrow_{M_0} v_0$ and $M_0 \le N_0$, where $p_0 = e\{\vec{u}/\vec{x}\}$, $\vec{x} \vdash e\ R^{\mathsf{C}}\ e'$ and $\vdash \vec{u}\ \overline{S}\ \vec{u}'$ such that $\vdash e\{\vec{u}/\vec{x}\}\ R^{\mathsf{C}}\{\overline{S}\}\ e'\{\vec{u}'/\vec{x}\}$. We will show $e'\{\vec{u}'/\vec{x}\} \Downarrow v_0'$, for some v_0' such that $\vdash v_0\ \overline{S}\ v_0'$.

Consider the derivation of $\vec{x} \vdash e\ R^{\mathsf{C}}\ e'$. There are two cases.

(Ctx R) Suppose $\vec{x} \vdash e \ R \ e'$. Since $\vdash \vec{u} \ \overline{S} \ \vec{u}'$, there exist $\vec{w}$ and $\vec{w}'$ such that $\vdash \vec{u} \ \overline{\gtrsim^\circ} \ \vec{w} \ \overline{R^{\mathsf{C}}} \ \vec{w}' \ \overline{\lesssim^\circ} \ \vec{u}'$. As $\gtrsim^\circ$ and $\lesssim^\circ$ are precongruences, we get

$$e\{\vec{u}/\vec{x}\} \gtrsim e\{\vec{w}/\vec{x}\} \ R\{\overline{R^{\mathsf{C}}}\} \ e'\{\vec{w}'/\vec{x}\} \lesssim e'\{\vec{u}'/\vec{x}\}$$

Since $e\{\vec{u}/\vec{x}\} \Downarrow v_0$, also $e\{\vec{w}/\vec{x}\} \Downarrow v$ with $\vdash v_0 \ \overline{\gtrsim^\circ} \ v$. From assumption $R\{\overline{R^{\mathsf{C}}}\} \cap Rel \subseteq \langle S \rangle$ follows $e'\{\vec{w}'/\vec{x}\} \Downarrow v'$ with $\vdash v \ \overline{S} \ v'$. Finally, $e'\{\vec{u}'/\vec{x}\} \Downarrow v_0'$ with $\vdash v' \ \overline{\lesssim^\circ} \ v_0'$, and $\vdash v_0 \ \overline{\gtrsim^\circ} \ v \ \overline{S} \ v' \ \overline{\lesssim^\circ} \ v_0'$ implies $\vdash v_0 \ \overline{S} \ v_0'$ because $\gtrsim^\circ S \lesssim^\circ \subseteq S$ by transitivity of $\gtrsim^\circ$ and $\lesssim^\circ$.

(Ctx Comp) If $\vec{x} \vdash e \ \widehat{R^{\mathsf{C}}} \ e'$, we argue as in the proof of Lemma 6 above (with R^{C} and $\overline{S}$ in place of S and $\overline{S+}$), except that here we do not keep track of the cost of the evaluation $e'\{\vec{u}'/\vec{x}\} \Downarrow v_0'$. Again the result is immediate if e is a variable. Otherwise we proceed by analysis of the derivation of $e\{\vec{u}/\vec{x}\} \Downarrow_{M_0} v_0$. Only the cases when this is derived by (Eval fn) or (Eval apply) are different from those in the proof of Lemma 6.

Case (Eval fn) $e\{\vec{u}/\vec{x}\}$ is a function and $e\{\vec{u}/\vec{x}\} = v_0$. Since e is not a variable, it is itself a function, $e = \texttt{fn}\ y \texttt{ => } d$ for some $d \in Exp_{\vec{x}y}$. Then $\vec{x} \vdash e \ \widehat{R^{\mathsf{C}}} \ e'$ must be derived by (Comp fn) so that $e' = \texttt{fn}\ y \texttt{ => } d'$ where $\vec{x}y \vdash d \ R^{\mathsf{C}} \ d'$. Since $\vdash \vec{u} \ \overline{S} \ \vec{u}'$, there exist $\vec{w}$ and $\vec{w}'$ such that $\vdash \vec{u} \ \overline{\gtrsim^\circ} \ \vec{w} \ \overline{R^{\mathsf{C}}} \ \vec{w}' \ \overline{\lesssim^\circ} \ \vec{u}'$. As $\gtrsim^\circ$, R^{C}, and $\lesssim^\circ$ are compatible, we get

$$y \vdash d\{\vec{u}/\vec{x}\} \gtrsim^\circ d\{\vec{w}/\vec{x}\} \ R^{\mathsf{C}}\{\overline{R^{\mathsf{C}}}\} \ d'\{\vec{w}'/\vec{x}\} \lesssim^\circ d'\{\vec{u}'/\vec{x}\}.$$

Let $(let\ \vec{x} = \vec{v}\ in\ e)$ abbreviate a suitable `let` construction which beta reduces to $e\{\vec{v}/\vec{x}\}$. Then

$$y \vdash d\{\vec{w}/\vec{x}\} \ \underset{\sim}{\diamond}^\circ \ (let\ \vec{x} = \vec{w}\ in\ d) \ R^{\mathsf{C}} \ (let\ \vec{x} = \vec{w}'\ in\ d') \ \underset{\sim}{\diamond}^\circ \ d'\{\vec{w}'/\vec{x}\}.$$

Since $\gtrsim$ and $\lesssim$ contain $\underset{\sim}{\diamond}$ and are transitive,

$$y \vdash d\{\vec{u}/\vec{x}\} \ S \ d'\{\vec{u}'/\vec{x}\},$$

Hence $\vdash e\{\vec{u}/\vec{x}\} \ \overline{S} \ e'\{\vec{u}'/\vec{x}\}$, as required.

Case (Eval apply) $e = e_1 e_2$, $e_1\{\vec{u}/\vec{x}\} \Downarrow_{M_1} v_1 = \texttt{fn}\ y \texttt{ => } d_1$, $e_2\{\vec{u}/\vec{x}\} \Downarrow_{M_2} v_2$, $d_1\{v_2/y\} \Downarrow_{M_3} v_0$, and $M_0 = M_1 + M_2 + M_3 + 1$. Since $\vec{x} \vdash e \ \widehat{R^{\mathsf{C}}} \ e'$, $e' = e_1' e_2'$ with $\vec{x} \vdash e_i \ R^{\mathsf{C}} \ e_i'$. We observe that $M_i < N_0$ because $M_i < M$ and $M \leq N_0$, for $i = 1, 2, 3$. By induction hypothesis (I.H.2), $e_i\{\vec{u}/\vec{x}\} \Downarrow_{M_i} v_i$ implies $e_i'\{\vec{u}'/\vec{x}\} \Downarrow v_i'$ with $\vdash v_i \ \overline{S} \ v_i'$, for $i = 1, 2$. So $v_1' = \texttt{fn}\ y \texttt{ => } d_1'$ where $y \vdash d_1 \ S \ d_1'$, that is, there exist $d, d' \in Exp_y$ such that $y \vdash d_1 \gtrsim^\circ d \ R^{\mathsf{C}} \ d' \lesssim^\circ d_1'$. Since $\gtrsim^\circ$ and $\lesssim^\circ$ are closed under substitutions, $d_1\{v_2/y\} \gtrsim d\{v_2/y\}$ and $d'\{v_2'/y\} \lesssim d_1'\{v_2'/y\}$. From the former and $d_1\{v_2/y\} \Downarrow_{M_3} v_0$, we get $d\{v_2/y\} \Downarrow_{\leq M_3} v$ with $\vdash v_0 \ \overline{\gtrsim^\circ} \ v$. As $M_3 < N_0$ and $\vdash d\{v_2/y\} \ R^{\mathsf{C}}\{\overline{S}\} \ d'\{v_2'/y\}$, induction hypothesis (I.H.1) and $d\{^{v_2}\!/_{y}\} \Downarrow_{\leq M_3} v$ imply $d'\{^{v_2'}\!/_{y}\} \Downarrow v'$ with $\vdash v \ \overline{S} \ v'$. Since $d'\{^{v_2'}\!/_{y}\} \lesssim d_1'\{^{v_2'}\!/_{y}\}$, $d_1'\{^{v_2'}\!/_{y}\} \Downarrow v_0'$ with $\vdash v' \ \overline{\lesssim^\circ} \ v_0'$. Then $\vdash v_0 \ \overline{S} \ v_0'$ because $\overline{\underset{\sim}{\rhd}^\circ} \ \overline{S} \ \overline{\lesssim^\circ} = \overline{\underset{\sim}{\rhd}^\circ \ S \lesssim^\circ}$ and $\underset{\sim}{\rhd}^\circ \ S \lesssim^\circ \ \subseteq S$. By (Eval apply), we conclude $e'\{^{\vec{u}'}\!/_{\vec{x}}\} \Downarrow v_0'$.

This establishes $R^{\mathsf{C}}\{\overline{S}\} \subseteq \langle S \rangle$ and then (A.2) follows, as required. □

References

Abramsky, S. (1990). The lazy lambda calculus. In D. Turner (Ed.), *Research topics in functional programming*, pp. 65–116. Addison-Wesley.

Bird, R. and P. Wadler (1987). *Introduction to Functional Programming*. International Series in Computer Science. Prentice-Hall.

Braüner, T. (1996, November). An axiomatic approach to adequacy. Technical Report DS-96-4, BRICS, Department of Computer Science, University of Aarhus. Ph.D. thesis.

Clinger, W. and J. Rees (editors) (1991, July-September). Revised^4 report on the algorithmic language scheme. *LISP Pointers IV*(3), 1–55.

Felleisen, M. and D. P. Friedman (1987). Control operators, the SECD-machine, and the λ-calculus. In M. Wirsing (Ed.), *Formal Description of Programming Concepts III*. IFIP.

Ferreira, W., M. Hennessy, and A. Jeffrey (1995, September). A theory of weak bisimulation for core CML. Technical Report 05/95, COGS, University of Sussex.

Gordon, A. D. (1994). *Functional Programming and Input/Output*. Cambridge University Press.

Gordon, A. D. (1995, July). Bisimilarity as a theory of functional programming. Mini-course. BRICS Notes Series NS-95-3, BRICS, Department of Computer Science, University of Aarhus.

Gordon, A. D. (1997). Operational equivalences for untyped and polymorphic object calculi. See Gordon and Pitts (1997).

Gordon, A. D. and A. M. Pitts (Eds.) (1997). *Higher Order Operational Techniques in Semantics*, Publications of the Newton Institute. Cambridge University Press.

Gordon, A. D. and G. D. Rees (1996). Bisimilarity for a first-order calculus of objects with subtyping. In *POPL'96 Symposium on Principles of Programming Languages*. ACM.

Howe, D. J. (1989). Equality in lazy computation systems. In *4th Annual Symposium on logic in computer science*. IEEE.

Howe, D. J. (1996). Proving congruence of bisimulation in functional programming languages. *Information and Computation 124*(2), 103–112.

Klop, J. W., V. van Oostrom, and F. van Raamsdonk (1993). Combinatory reduction systems: introduction and survey. *Theoretical Computer Science 121*, 279–308.

Lassen, S. B. (1997). Action semantics reasoning about functional programs. *Mathematical Structures in Computer Science*. Special issue dedicated to the Workshop on Logic, Domains, and Programming Languages (Darmstadt, May 1995). To appear.

Lawall, J. L. and H. G. Mairson (1996). Optimality and inefficiency: what isn't a cost model of the lambda calculus. In *ICFP'96 International Conference on Functional Programming*, pp. 92–101. ACM.

Mason, I. A., S. F. Smith, and C. L. Talcott (1996). From operational semantics to domain theory. *Information and Computation 128*(1), 26–47.

Milner, R. (1977). Fully abstract models of typed lambda-calculi. *Theoretical Computer Science 4*, 1–23.

Milner, R. (1989). *Communication and Concurrency*. Prentice-Hall.

Milner, R., M. Tofte, and R. Harper (1990). *The Definition of Standard ML*. Cambridge, Mass.: MIT Press.

Moggi, E. (1989). Computational lambda-calculus and monads. In *4th Annual Symposium on logic in computer science*, pp. 14–23. IEEE.

Ong, C.-H. L. (1992, August 7). Concurrent lambda calculus and a general pre-congruence theorem for applicative bisimulation (preliminary version). Unpublished.

Park, D. M. (1981). Concurrency and automata on infinite sequences. In P. Deussen (Ed.), *Conference on Theoretical Computer Science*, Volume 104 of *Lecture Notes in Computer Science*, pp. 167–183. Springer-Verlag.

Pitts, A. M. (1994a, November). Inductive and co-inductive techniques in the semantics of functional programs. Course held at BRICS, Department of Computer Science, University of Aarhus.

Pitts, A. M. (1994b, December). Some notes on inductive and co-inductive techniques in the semantics of functional programs (draft version). BRICS Notes Series NS-94-5, BRICS, Department of Computer Science, University of Aarhus.

Pitts, A. M. (1995, March). An extension of Howe's construction to yield simulation-up-to-context results. Unpublished Manuscript.

Pitts, A. M. (1996a). A note on logical relations between semantics and syntax. Submitted to the Journal of the Interest Group in Pure and Applied Logics, Special Issue for 3rd Workshop on Logic, Language, Information and Computation (WoLLIC'96) May, 1996 Salvador (Bahia), Brazil.

Pitts, A. M. (1996b). Relational properties of domains. *Information and Computation 127*, 66–90.

Pitts, A. M. (1997). Operationally-based theories of program equivalence. In P. Dybjer and A. M. Pitts (Eds.), *Semantics and Logics of Computation*. Cambridge University Press.

Sands, D. (1991). Operational theories of improvement in functional languages (extended abstract). In *Proceedings of the Fourth Glasgow Workshop on Functional Programming*, Workshops in Computing Series, pp. 298–311. Springer-Verlag.

Sands, D. (1997a). From SOS rules to proof principles: An operational metatheory for functional languages. In *POPL'97 Symposium on Principles of Programming Languages*. ACM.

Sands, D. (1997b). Improvement theory and its applications. See Gordon and Pitts (1997).

Sangiorgi, D. (1994, August). On the bisimulation proof method. Technical Report LFCS-94-299, University of Edinburgh.

Sangiorgi, D. (1995, April). Lazy functions and mobile processes. Technical Report RR-2515, INRIA-Sophia Antipolis.

Sangiorgi, D. (1996). Locality and interleaving semantics in calculi for mobile processes. *Theoretical Computer Science 155*(1), 39–83.

Sangiorgi, D. and R. Milner (1992). The problem of "weak bisimulation up to". In W. R. Cleveland (Ed.), *CONCUR '92*, Volume 630 of *Lecture Notes in Computer Science*. Springer-Verlag.

Smith, S. F. (1997). The coverage of operational semantics. See Gordon and Pitts (1997).

Stoughton, A. (1988). Substitution revisited. *Theoretical Computer Science 59*, 317–325.

Talcott, C. (1997). Reasoning about functions with effects. See Gordon and Pitts (1997).

Winskel, G. (1993). *The Formal Semantics of Programming Languages*. Cambridge, Mass.: MIT Press.

Recent BRICS Report Series Publications

RS-97-24 **Søren B. Lassen.** ***Relational Reasoning about Contexts*****. September 1997. 45 pp. To appear as a chapter in the book** ***Higher Order Operational Techniques in Semantics*****, eds. Andrew D. Gordon and Andrew M. Pitts, Cambridge University Press.**

RS-97-23 **Ulrich Kohlenbach.** ***On the Arithmetical Content of Restricted Forms of Comprehension, Choice and General Uniform Boundedness*****. August 1997. 35 pp.**

RS-97-22 **Carsten Butz.** ***Syntax and Semantics of the logic*** $\mathcal{L}^{\lambda}_{\omega\omega}$**. July 1997. 14 pp.**

RS-97-21 **Steve Awodey and Carsten Butz.** ***Topological Completeness for Higher-Order Logic*****. July 1997. 19 pp.**

RS-97-20 **Carsten Butz and Peter T. Johnstone.** ***Classifying Toposes for First Order Theories*****. July 1997. 34 pp.**

RS-97-19 **Andrew D. Gordon, Paul D. Hankin, and Søren B. Lassen.** ***Compilation and Equivalence of Imperative Objects*****. July 1997. iv+64 pp. Appears also as Technical Report 429, University of Cambridge Computer Laboratory, June 1997. To appear in** ***Foundations of Software Technology and Theoretical Computer Science: 17th Conference*****, FCT&TCS '97 Proceedings, LNCS, 1997.**

RS-97-18 **Robert Pollack.** ***How to Believe a Machine-Checked Proof*****. July 1997. 18 pp. To appear as a chapter in the book** ***Twenty Five Years of Constructive Type Theory*****, eds. Smith and Sambin, Oxford University Press.**

RS-97-17 **Peter Bro Miltersen.** ***Error Correcting Codes, Perfect Hashing Circuits, and Deterministic Dynamic Dictionaries*****. June 1997. 10 pp.**

RS-97-16 **Noga Alon, Martin Dietzfelbinger, Peter Bro Miltersen, Erez Petrank, and Gábor Tardos.** ***Linear Hashing*****. June 1997. 22 pp. A preliminary version appeared with the title** ***Is Linear Hashing Good?*** **in** ***The Twenty-ninth Annual ACM Symposium on Theory of Computing*****, STOC '97, pages 465–474.**

RS-97-15 **Pierre-Louis Curien, Gordon Plotkin, and Glynn Winskel.** ***Bistructures, Bidomains and Linear Logic*****. June 1997. 41 pp.**