On the representation of McCarthy's *amb* in the π-calculus

Arnaud Carayol[1,2], Daniel Hirschkoff[2], and Davide Sangiorgi[3]

[1] IRISA - Rennes, France
[2] LIP - ENS Lyon, France
[3] University of Bologna, Italy

Abstract. We study the encoding of $\lambda^{\talloblong}$, the call by name λ-calculus enriched with McCarthy's *amb* operator, into the π-calculus. Semantically, *amb* is a challenging operator, for the fairness constraints that it expresses. We prove that, under a certain interpretation of divergence in the λ-calculus (*weak divergence*), a faithful encoding is impossible. However, with a different interpretation of divergence (*strong divergence*), the encoding is possible, and for this case we derive results and coinductive proof methods to reason about $\lambda^{\talloblong}$ that are similar to those for the encoding of pure λ-calculi. We then use these methods to derive the most important laws concerning *amb*. We take bisimilarity as behavioural equivalence on the π-calculus, which sheds some light on the relationship between fairness and bisimilarity. As a spin-off result, we show that there is no small-step operational semantics for $\lambda^{\talloblong}$ that preserves the branching structure of terms and yields the correct predicates of convergence and (weak) divergence.

1 Introduction

The operator of ambiguous choice, *amb*, was first introduced in [McC63], to describe a form of composition of (partial) functions that is liable to return one among several results. [McC63] describes *amb* by giving its main properties. The two most important properties have to do with fairness. One property says that *amb* is *bottom-avoiding*, meaning that the composition of a function with a function that is undefined should return the result of the former function. The other important property says that *amb* behaves as a non-deterministic choice whenever the results computed by the functions being composed are both defined: either of them may be returned, in an unpredictable way. The usefulness for an operator having the properties of *amb* has come to light for the specification of systems, in particular operating systems, essentially because a form of fair non-determinism is required to merge incoming messages (see [Hen82,Tur90], and also [HO90], that studies *amb* and other nondeterministic operators with respect to this issue). The main reason, however, for our interest in *amb* is that, semantically, 40 years later, *amb* remains a very challenging operator [Las98,Mor98,LM99,Pit01,FK02].

The difficulties introduced by *amb* are clear in $\lambda^{\talloblong}$, the call-by-name λ-calculus enriched with the binary operator $\talloblong$ that is a 'realisation' of McCarthy's *amb*. The two standard approaches to obtaining semantics and analysis techniques for λ-calculi are the denotational and the operational ones. The former is based on domain theory; in the latter, applicative bisimilarity is exploited to reason about contextual equivalence. The problem for denotational analyses is that *amb* is not continuous (see [Mor98] for a discussion). The operational approach has been followed by Moran, Lassen and Pitcher, in a series of works [Las98,Mor98,LM99,Pit01]. The problem of proving congruence of applicative bisimilarity (or a similar coinductively defined relation, that coincides with or at least gives a good approximation of contextual equivalence) is however still open for $\lambda^{\talloblong}$. The usual technique for proving congruence of applicative bisimilarity in λ-calculi is Howe's [How96], but this technique does not seem to work in presence of *amb* [LM99]. Therefore, to prove a set of characteristic laws of *amb*, some 'partial' proof techniques have been developed, in particular in [Mor98,LM99] (these techniques are *partial* in the sense that, taken separately, none of them can be used to derive all the laws – see also Section 5). It would be very hard and tedious to prove the laws following the definition of contextual equivalence, due to its heavy quantification on contexts.

In the present paper, we explore an alternative way to give the semantics of $\lambda^{\talloblong}$, via an encoding into the (asynchronous) π-calculus. There were various reasons for carrying out this study. The first reason is the quest for proof methods to reason about languages like $\lambda^{\talloblong}$ that contain operators expressing fairness constraints. The problem of encoding the λ-calculus (as well as parallel and nondeterministic extensions of it) into the π-calculus has been extensively studied – see e.g. [Mil90,San92,BL00,SW01]. In the case of the call-by-name λ-calculus, for example, the π-calculus semantics induces an equivalence on λ-terms that coincides with the classical Lévy-Longo Tree semantics [SW01], which shows an agreement between the π-calculus semantics and standard denotational analyses of the call-by-name λ-calculus. Moreover, bisimulation is the canonical equivalence in the π-calculus, and comes with a well-developed theory, as well as powerful proof techniques that alleviate the task of building bisimulation proofs. One can therefore hope that working in the π-calculus can help in defining useful bisimulation-based techniques for $\lambda^{\talloblong}$.

A second motivation for this study is expressiveness. The π-calculus has been shown to be a very powerful formalism. We want to understand whether, and under which conditions, the π-calculus can encode an operator as sophisticated as *amb*. We are not aware of other attempts at encoding into the π-calculus operators that express fairness constraints.

Another motivation is the question of fairness in the π-calculus. While the standard SOS rules of the π-calculus make no reference to fairness, the use of bisimulation or of similar semantical equivalences introduces this kind of property. The definition of a semantics for a fair operator like *amb* is a way to gain a better understanding of this issue. To illustrate this point, consider the π-calculus term $\tau^{\omega} \,|\, \overline{a}$, where τ^{ω} represents a process that can perform infinitely many internal actions, $\overline{a}$ is an output at a without value exchange, and "$|$" is the operator of parallel composition. Under bisimulation equivalence, as opposed to, say, testing equivalence, this process is deemed the same as the process $\overline{a}$. One way of interpreting this equality is to say that bisimilarity ignores divergence. However, another way of looking at the equality is to say that bisimilarity encompasses some fairness: under a fair implementation of parallel composition, the left component τ^{ω} cannot always prevail, hence eventually the action $\overline{a}$ on the right-hand side will be executed. It is precisely this second – and usually neglected – interpretation of bisimilarity that we are addressing, trying to understand its significance on a non-trivial concrete example.

When studying non-deterministic operators like *amb*, contextual equivalence is defined by observing the ability for two terms, in any context, to exhibit convergences and divergences. Two kinds of divergences can be distinguished (see e.g. [NC95]): a computation in which convergence is impossible is a *strong divergence*, while a *weak divergence* corresponds to an infinite computation along which the possibility to converge to a value is never lost. Both forms of divergence arise in $\lambda^{\talloblong}$: first notice that Ω, the usual always diverging term, is strongly divergent. To give an example of a weak divergence, we use the operator of *internal choice*, $\oplus$, that can be encoded in $\lambda^{\talloblong}$ as follows:

$$M \oplus N \stackrel{\mathsf{def}}{=} (K\, M \talloblong K\, N)\, I\,,$$

K and I being the usual combinators for selection and identity. By definition of $\lambda^{\talloblong}$, $M \oplus N$

can nondeterministically evolve to M or N. Now consider the term

$$T \overset{\mathsf{def}}{=} \mathsf{Fix}\, \lambda x.\, (x \oplus I)$$

(where Fix is defined as $A\,A$, with $A \overset{\mathsf{def}}{=} \lambda xy.\, y\ (x\ x\ y)$). Because of the 'erratic' nature of internal choice, T exhibits a weak divergence, along which convergence to I is repeatedly discarded. In the operational studies of *amb* in the literature, strong and weak divergences are not distinguished. In this paper, in contrast, we separate the case in which both strong and weak divergences count from the case in which only strong divergences count. Interestingly, the difference between strong and weak divergence does not affect the *characteristic laws* of *amb*: we refer here to a set of laws that capture *amb*'s essential properties (these laws are studied for example in [Mor98]) – as mentioned above, the original specification of *amb* [McC63] is given at a very high level by describing its behavioural properties.

The semantics of *amb* is very challenging also in the π-calculus. Indeed we prove that, under the "weak" interpretation of divergence in $\lambda^{[\!]}$, a faithful encoding into the π-calculus is impossible. By 'faithful' we mean that the encoding should be sound and should mimic the behaviour of $\lambda^{[\!]}$ terms, at least as far as divergence and reduction to values are concerned. This result holds for the π-calculus, as well as any extension of π-calculus with *finitary* operators.

However, we also show that with the strong interpretation of divergence, the encoding of $\lambda^{[\!]}$ into the π-calculus is possible, and we derive results and coinductive proof methods to reason about $\lambda^{[\!]}$ that are similar to those that have been developed for the encodings of pure λ-calculi (see [SW01]). We then use these methods to derive the characteristic laws of McCarthy's *amb*. Using the π-calculus proof techniques, the proof of some of these laws is very simple, in particular that of the two key laws of *amb*, the bottom-avoidance law $M \mathbin{[\!]} \Omega \;\cong_{\mathrm{M}}\; M$, and the law $V \mathbin{[\!]} V' \;\cong_{\mathrm{M}}\; V \oplus V'$ (where V and V' are λ-abstractions). We also study the extension of $\lambda^{[\!]}$ with local call-by-value, again showing an encoding into the π-calculus and then using the encoding for deriving algebraic laws of the source calculus.

We have mentioned above that a characterisation of operational equivalence in $\lambda^{[\!]}$ as a form of applicative bisimilarity is still an open problem. As a spin-off result of our study suggests that it might indeed be very hard to obtain such a bisimilarity. The result says that there is no small-step operational semantics for $\lambda^{[\!]}$ that preserves the branching structure of terms and yields the correct predicates of convergence and divergence. An operational semantics having these properties is important for defining bisimilarity in languages with non-determinism like $\lambda^{[\!]}$, for bisimulation is based on the branching structure of terms and to be really useful in practice, in the bisimulation game between two non-deterministic terms, the challenge should use the small-step semantics.

Outline. After recalling the necessary definitions and results we need about $\lambda^{[\!]}$ and the π-calculus (Section 2), Section 3 presents our semantic framework for McCarthy's *amb*, as well as some results for it that motivate the study in the following sections. In Sections 4 to 6, we introduce our π-calculus encoding of $\lambda^{[\!]}$, and present a number of applications and developments of it. We conclude and discuss further research directions in Section 7. Detailed proofs can be found in [CHS03].

2 Background

This section contains background material. (However it also contains some novel results: a new semantics for $\lambda^{[\!]}$ and some new up-to proof techniques for coupled simulation.)

$$\textsc{Beta}\ (\lambda x.\, M)\ N \ \twoheadrightarrow\ M[N/x]$$

$$\textsc{Lazy}\ \frac{M \ \twoheadrightarrow\ M'}{M\, N \ \twoheadrightarrow\ M'\, N} \qquad \textsc{Trans}\ \frac{M \ \twoheadrightarrow\ M' \qquad M' \ \twoheadrightarrow\ M''}{M \ \twoheadrightarrow\ M''}$$

$$\textsc{Par}\ \frac{M \ \twoheadrightarrow\ M' \qquad N \ \twoheadrightarrow\ N'}{M \,[]\, N \ \twoheadrightarrow\ M' \,[]\, N'} \quad \textsc{Val}_L\ \frac{M \ \twoheadrightarrow\ V \quad \textbf{or} \quad M \ =\ V}{M \,[]\, N \ \twoheadrightarrow\ V}$$

Fig. 1. Operational semantics for $\lambda^{[]}$

2.1 The λ-calculus with Ambiguous Choice

We suppose we have an infinite set of *variables*, ranged over with $x, y, \ldots$. *Terms* of $\lambda^{[]}$, ranged over with $M, N, \ldots$, are given by the following grammar:

$$M \quad \overset{\mathsf{def}}{=} \quad x \mid \lambda x.M \mid M_1\, M_2 \mid M_1 \,[]\, M_2\,.$$

Bound and free variables are defined as usual. A *closed term* is a term that contains no free variable. Substitution (written $M[N/x]$) and α-conversion are defined as usual. We will work up-to α-conversion. Closed values, ranged over with $V, V', \ldots$, are abstractions. A context, ranged over with $C, C', \ldots$ is a term containing occurrences of a *hole*, written $[\,\cdot\,]$, in it. Given a context C, $C[M]$ denotes the term obtained by replacing the hole with a term M in C. Given M, C is *closing* if $C[M]$ is closed, this terminology being extended to the case where C 'closes' several terms.

Here are some $\lambda^{[]}$ terms, that will be useful below:

$$I \overset{\mathsf{def}}{=} \lambda x.\, x \qquad \Omega \overset{\mathsf{def}}{=} (\lambda x.\, x\, x)\ (\lambda x.\, x\, x)$$
$$K \overset{\mathsf{def}}{=} \lambda x\, y.\, x \qquad \mathsf{Fix} \overset{\mathsf{def}}{=} A\, A \ \text{ where } \ A \overset{\mathsf{def}}{=} \lambda x\, y.\, y\, (x\, x\, y)\,.$$

We introduce some notations for relations:

Definition 1 (Relations). *If $\mathcal{R}$ is a binary relation over elements of a set $\mathbb{S}$, $\mathcal{R}^{-1}$ denotes the inverse of $\mathcal{R}$, while $\mathcal{R}^{+}$ and $\mathcal{R}^{*}$ denote the transitive (resp. transitive and reflexive) closures of $\mathcal{R}$. Composition of two relations $\mathcal{R}$ and $\mathcal{S}$ is written $\mathcal{R}\mathcal{S}$, and $T\,\mathcal{R}^{\omega}$ stands for the existence of an infinite sequence of elements of $\mathbb{S}$, $T = T_0, T_1, \ldots$ such that for all i, $T_i\, \mathcal{R}\, T_{i+1}$.*

Definition 2 ($\twoheadrightarrow$). *Relation $\twoheadrightarrow$ is given by the rules of Figure 1, where the symmetrical version of* $\textsc{Val}_L$ *is omitted.*

In defining $\twoheadrightarrow$, we capture the transitive, *non-reflexive* closure of the underlying reduction relation. In rule $\textsc{Par}$ both components of an *amb* are allowed to evolve. Rules $\textsc{Val}_L$, $\textsc{Val}_R$ make the choice between components of an *amb*, when one of the branches converges.

Remark 1. The semantics $\twoheadrightarrow$ of Figure 1 is new. It coincides with the semantics $\rightsquigarrow$ introduced in [Mor98] (see [CHS03]) but is defined directly on $\lambda^{[]}$-terms whereas the definition of $\rightsquigarrow$ requires the use of decorated terms.

Definition 3. *[$\Downarrow$ and $\Uparrow$] A term M is* convergent*, written $M \Downarrow$, if there exists a value V s.t. $M \twoheadrightarrow V$. M is* divergent*, written $M \Uparrow$, if $M \twoheadrightarrow^{\omega}$.*

We can remark that by definition of $\twoheadrightarrow$, a term of the form $M \,[]\, I$, for any M, cannot diverge: this observation will be useful in several proofs below.

Definition 4 (Observational equivalence, using divergence). *M and N are* observationally equivalent*, written $M \cong_{M} N$, iff for any closing context C:*

$$(C[M]\Downarrow \iff C[N]\Downarrow) \quad \textit{and} \quad (C[M]\Uparrow \iff C[N]\Uparrow)\,.$$

2.2 The Asynchronous π-calculus

We suppose that we have an infinite set of *names*, also called *channels*, over which we range with small letters: $a, b, \ldots, x, y, \ldots$. For the sake of the Asynchronous π-calculus (in short, $A\pi$) encoding of Section 4, we shall translate a $\lambda^{\parallel}$ variable using a π-calculus name, and we suppose that there is an injection from variables to names so that we can keep letter x to refer to the encoding of a variable x. (Possibly empty) name tuples are ranged over with $\tilde{x}, \tilde{y}, \ldots$. $A\pi$ terms, to which we shall refer simply as *processes*, are ranged over using $P, Q, \ldots$, and are defined as follows:

$$P \;\overset{\mathsf{def}}{=}\; \mathbf{0} \mid P_1|P_2 \mid !P \mid \boldsymbol{\nu} x\; P \mid x(\tilde{y}).P \mid \bar{x}\langle\tilde{y}\rangle\,.$$

Our calculus has the inactive process ($\mathbf{0}$), parallel composition, replication, restriction, the input prefix and asynchronous output. Bound names in processes are defined by saying that the input and restriction operators are binding. Contexts in $A\pi$ are defined along the lines of $\lambda^{\parallel}$ contexts.

Late operational semantics for $A\pi$ is introduced as usual. It defines judgements of the form $P \xrightarrow{\mu} P'$, where μ is either an *input action* of the form $a(\tilde{x})$, a (bound) *output action* of the form $\boldsymbol{\nu}\tilde{x}\; \bar{a}\langle\tilde{y}\rangle$, in which $\tilde{x}$ has a set (instead of a tuple) structure and all names of $\tilde{x}$ occur in $\tilde{y}$, or τ, which denotes internal computation. We introduce the following notations: $\Rightarrow \overset{\mathsf{def}}{=} (\xrightarrow{\tau})^*$, $\xrightarrow{\hat{\mu}} \overset{\mathsf{def}}{=} \xrightarrow{\tau}$ or $=$ if $\mu = \tau$, $\xrightarrow{\hat{\mu}} \overset{\mathsf{def}}{=} \xrightarrow{\mu}$ otherwise, and $\overset{\hat{\mu}}{\Longrightarrow} \overset{\mathsf{def}}{=} \Rightarrow\xrightarrow{\hat{\mu}}\Rightarrow$.

Structural congruence, $\equiv$, is also defined as usual, to capture some basic structural properties of processes. It is needed in the statement of the following result, which will be useful for a proof below:

Proposition 1 ($\xrightarrow{\tau}_{/\equiv}$ is finitely branching). *Given a process P, there is, up to structural congruence, a finite number of processes P' such that $P \xrightarrow{\tau} P'$.*

We shall use the following behavioural relations on processes:

Definition 5 (Behavioural equivalences and preorders, $\approx, \leftrightarrows$).

- *A binary relation $\mathcal{R}$ on processes is a* weak simulation *if $P\,\mathcal{R}\,Q$ and $P \xrightarrow{\mu} P'$ imply that there exists Q' such that $Q \overset{\hat{\mu}}{\Longrightarrow} Q'$ and $P'\,\mathcal{R}\,Q'$.*
- *A weak bisimulation is a symmetric weak simulation.* Weak bisimilarity*, written $\approx$, is the greatest weak bisimulation.*
- *A* coupled bisimulation *is a pair of simulations $(\mathcal{S}_1, \mathcal{S}_2^{-1})$ such that:*

- if $P\,\mathcal{S}_1\,Q$ then there exists Q' s.t. $Q \Rightarrow Q'$ and $P\,\mathcal{S}_2\,Q'$;
- if $P\,\mathcal{S}_2\,Q$ then there exists P' s.t. $P \Rightarrow P'$ and $P'\,\mathcal{S}_1\,Q$.
Two processes P and Q are coupled bisimilar*, written $P \leftrightarrows Q$, if there exists a coupled bisimulation $(\mathcal{S}_1, \mathcal{S}_2^{-1})$ such that $P\,\mathcal{S}_1\,Q$ and $P\,\mathcal{S}_2\,Q$.*

Coupled bisimulation has been used for the π-calculus e.g. in [NP96].

Definition 6 ($\cong_\pi$). *Given a name p, $P \Downarrow_p$ stands for $P \Rightarrow\xrightarrow{\boldsymbol{\nu}\tilde{x}\;\bar{p}\langle\tilde{y}\rangle}$ for some $\tilde{x}$ and $\tilde{y}$. P and Q are* observationally equivalent*, written $P \cong_\pi Q$, iff*

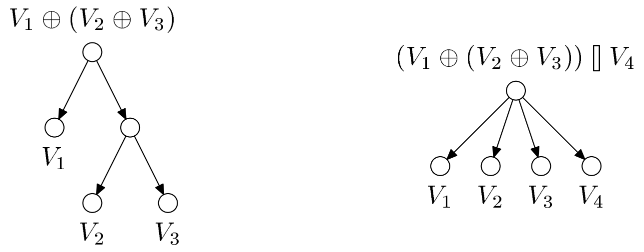

Fig. 2. Derivations trees of $V_1 \oplus (V_2 \oplus V_3)$ and $V_1 \oplus (V_2 \oplus V_3) \, [\,] \, V_4$ for $\twoheadrightarrow$

$$(\textit{for all } C \textit{ and } p, \quad C[P] \Downarrow_p \;\Leftrightarrow\; C[Q] \Downarrow_p) \quad \textit{and} \quad (P \Rightarrow\approx \mathbf{0} \;\Leftrightarrow\; Q \Rightarrow\approx \mathbf{0}) .$$

The definition of $\cong_\pi$ follows the pattern of $\cong_{\mathrm{M}}$ in $\lambda^{[\,]}$ (Definition 4, see also Definition 9 below). In $A\pi$, observables are output particles, and visible (strong) divergences, arising from terms that are compelled to diverge, equate such terms with $\mathbf{0}$.

Proposition 2 (Congruence properties). *$\approx$ and $\leftrightarrows$ are congruences in $A\pi$.*

We have $\approx \subseteq \leftrightarrows$. Moreover, $\approx \subseteq \cong_\pi$ and $\leftrightarrows \subseteq \cong_\pi$, and we shall use both $\approx$ and $\leftrightarrows$ to establish properties of $\cong_\pi$. This task will be made easy by the use of *up-to techniques*, essentially *up to context* and *up to expansion*. Such techniques are well-known for $\approx$ ([SW01]). We have proved that they can be adapted to $\leftrightarrows$ (we omit the details for lack of space, see [CHS03]).

3 Equivalence in $\lambda^{[\,]}$ Revisited

We now present our semantic framework for McCarthy's *amb* operator, including the distinction between strong and weak divergence. We study operational accounts of $\lambda^{[\,]}$, and introduce an equivalence based on strong divergences. We also show that if also weak divergences are taken into account in the notion of divergence, then a faithful encoding into the π-calculus is impossible.

3.1 Operational descriptions of *amb*

Let us examine how *amb* is described by $\twoheadrightarrow$. If we consider the terms given on Figure 2 (where the V_is are values), we see that according to $\twoheadrightarrow$, *amb* composition makes trees degenerate and loose their branching structure. Thus, $\twoheadrightarrow$ misses some choices along $\lambda^{[\,]}$ computations. This lack of precision can be seen as a drawback for defining a bisimulation-based equivalence for $\lambda^{[\,]}$, since such an equivalence usually exploits an accurate analysis of the decisions that are made along computation. Indeed, bisimulation equivalences are known to be more discriminating than trace equivalence, intuitively because they are based on trees and not on single executions (traces). In fact, on all terms of the form $M \, [\,] \, V$, $\twoheadrightarrow$ defines a big step semantics: such a term can only converge (immediately) to a value. $\twoheadrightarrow$ thus appears to be too imprecise to allow one to derive a suitable notion of bisimulation. One could then ask whether this could be improved, by providing a *branching preserving* presentation of a fair operational semantics for $\lambda^{[\,]}$, in the following sense:

Definition 7 (Branching preserving semantics). *We say that a relation* $\multimap$ *on* $\lambda^{\talloblong}$ *is a* branching preserving semantics *if for any reduction sequence* $M_0 \multimap \ldots \multimap M_n \multimap V$ *and any* $\lambda^{\talloblong}$*-term* P_0*, there exists a sequence* $(P_i)_{i\in[0,n]}$ *such that* $M_0 \talloblong P_0 \multimap^{+} \ldots \multimap^{+} M_n \talloblong P_n$.

This definition expresses the fact that the reduction tree of a term is preserved when this term is put in parallel with an other term. Standard small step semantics for a language that can express a form of parallel composition and nondeterministic choice usually have this property, as a consequence of their compositional nature. It turns out in this case that imposing this condition to a sufficiently accurate semantics of $\lambda^{\talloblong}$ prevents it to rule out forbidden divergences (according to *amb*'s specification).

Theorem 1 (No small step semantics preserving branching for $\lambda^{\talloblong}$). *There exists no branching preserving semantics for* $\lambda^{\talloblong}$ *that validates rules* BETA *and* LAZY *and induces the same notions of convergence and divergence as* $\twoheadrightarrow$.

This impossibility results therefore suggests that, even if some form of applicative bisimilarity for $\lambda^{\talloblong}$ were possible (i.e., definable, and provably a congruence), it might be of limited practical usefulness.

3.2 Strong vs Weak Divergences

We now study how working within the π-calculus affects the description of $\lambda^{\talloblong}$. By reasoning abstractly about 'reasonable' translations, we show that we cannot encode *amb* into the π-calculus if we demand to respect divergences (as defined in Definition 3):

Theorem 2 (No divergence-faithful encoding). *Let* $\Bumpeq$ *be an equivalence relation on* π*-calculus terms containing structural congruence. There does not exist an encoding* $[\![\cdot]\!]$ *of* $\lambda^{\talloblong}$ *in* $A\pi$ *such that, for any closed term* M*:*
(i) $[\![M]\!] \Bumpeq [\![N]\!] \;\Rightarrow\; M \cong_{M} N$ *(soundness w.r.t.* $\cong_{M}$*);*
(ii) $[\![M]\!] \xrightarrow{\tau}^{\omega} \Leftrightarrow\; M \twoheadrightarrow^{\omega}$ *(divergence faithfulness);*
(iii) $M \twoheadrightarrow V \;\Rightarrow\; [\![M]\!] \xrightarrow{\tau}^{+} \Bumpeq [\![V]\!]$ *(value preservation).*

We reason on terms such as

$$Z \overset{\mathsf{def}}{=} \mathsf{Fix}\,\lambda z.\ (I \talloblong (\lambda x.\, z\ (\lambda y.\, x))$$

Such a term can converge to $\lambda x_1 \ldots x_n.\, x_n$, for any $n \geq 1$, and cannot diverge. This entails, using clauses (*i*) and (*iii*), that the execution tree (modulo $\equiv$) of $[\![Z]\!]$ has infinitely many nodes. Using Proposition 1, we prove that this implies that this tree has an infinite branch. By clause (*ii*), this is in contradiction with the fact that Z cannot diverge.

Remark 2. The previous result holds in any *finitary* (i.e. preserving Proposition 1) extension of $A\pi$. To our knowledge, all extensions of the π-calculus considered in the literature are finitary, except for those including the operator of infinite sum.

As illustrated in Section 1, working with bisimulation in $A\pi$ leads us to distinguish between strong and weak divergences, that are defined as follows:

$$[\![\lambda x.M]\!]_p \overset{\mathsf{def}}{=} \boldsymbol{\nu} l\ (\bar{p}\langle l\rangle \mid l(x,q).[\![M]\!]_q) \qquad [\![x]\!]_p \overset{\mathsf{def}}{=} \boldsymbol{\nu} p'\ (\bar{x}\langle p'\rangle \mid p' \rightarrowtriangle p)$$

$$[\![M\,N]\!]_p \overset{\mathsf{def}}{=} \boldsymbol{\nu} q\ \left([\![M]\!]_q \mid q(l).\left(\boldsymbol{\nu} x\ \left(\bar{l}\langle x,p\rangle \mid !x(r).[\![N]\!]_r\right)\right)\right)$$

$$[\![M \,[]\, N]\!]_p \overset{\mathsf{def}}{=} \boldsymbol{\nu} p'\ ([\![M]\!]_{p'} \mid [\![N]\!]_{p'} \mid p' \rightarrowtriangle p) \qquad \text{where } q \rightarrowtriangle p \overset{\mathsf{def}}{=} q(x).\bar{p}\langle x\rangle$$

Fig. 3. Encoding of $\lambda^{[]}$ in the π-calculus

Definition 8 (Strong and weak divergences). *Let M be a $\lambda^{[]}$ term.*
- M is strongly divergent*, written $M \upharpoonleft$, if M can evolve into a term that cannot converge;*
- M is weakly divergent *if M exhibits an infinite computation along which it never loses the possibility to converge.*

A divergent term (Definition 3) is either strongly or weakly divergent, or both, as is $T \oplus \Omega$ (T is defined in Section 1).

We now adapt Definition 4 to focus on strong divergences.

Definition 9 ($\cong_\lambda$). *For any M, N, $M \cong_\lambda N$ if for any closing context C:*

$$(C[M] \Downarrow \ \Leftrightarrow\ C[N] \Downarrow) \quad \textit{and} \quad (C[M] \upharpoonleft \ \Leftrightarrow\ C[N] \upharpoonleft)\,.$$

Relations $\cong_\lambda$ and $\cong_{\mathrm{M}}$ (Definition 4) are incomparable: as $\cong_{\mathrm{M}}$ is sensitive to weak divergences, it separates terms that are equated by $\cong_\lambda$, hence $\cong_\lambda \not\subseteq \cong_{\mathrm{M}}$. Conversely, $\cong_\lambda \not\supseteq \cong_{\mathrm{M}}$ because $\cong_{\mathrm{M}}$ identifies weak and strong divergences. We have for instance:

$$I \quad \substack{\cong_\lambda \\ \not\cong_{\mathrm{M}}} \quad \mathsf{Fix}\ \lambda x.\,(x \oplus I) \quad \substack{\not\cong_\lambda \\ \cong_{\mathrm{M}}} \quad \Omega \oplus I\,.$$

Theorem 1 does not hold if only strong divergence counts, see [CHS03].

4 Encoding and soundness

Our encoding, written $[\![\_]\!]$, is defined on Figure 4, and follows the usual encodings of the λ-calculus for the operators of application and abstraction. A $\lambda^{[]}$-term M is mapped to a process $[\![M]\!]_p$, p being a channel where the value of (the encoding of) M will be passed (cf. the clause for abstraction).

To take $[]$ into account, we run the encodings of M and N in parallel at a freshly created location q, and let the (ephemeral) link process $q \rightarrowtriangle p$ forward any successfully terminated evaluation on p. Once $q \rightarrowtriangle p$ has been triggered by one of the components, the other component is isolated from the context, either because it tries to interact on the private channel q, or because it diverges. Modulo $\cong_\pi$, this corresponds to what we expect from *amb*.

Note the extra indirection $p' \rightarrowtriangle p$ in the encoding of variables. A similar indirection is needed in the encoding of call-by-value into (untyped) π-calculus, and can be removed using capability types [SW01]. We do not know how to avoid the indirection in the encoding of Figure 4 using types or other means.

Below is the soundness result for $[\![\_]\!]$. To prove it, we first establish operational correspondence, where each $\hookrightarrow$ reduction in $\lambda^{[]}$ is associated (up to *expansion*, a behavioural preorder related to weak bisimilarity) to a (non-empty) sequence of $\xrightarrow{\tau}$ reduction steps in the encoding π-calculus terms.

$$\begin{array}{ccc} M \talloblong N \cong_\lambda N \talloblong M & (M \talloblong N) \talloblong P \cong_\lambda M \talloblong (N \talloblong P) & (\lambda x.\, M)\, N \cong_\lambda M[N/x] \\ \multicolumn{2}{c}{M \talloblong \Omega \;\cong_\lambda\; M} & V \oplus V' \;\cong_\lambda\; V \talloblong V' \\ \multicolumn{2}{c}{(M \oplus N) \oplus P \;\cong_\lambda\; M \oplus (N \oplus P)} & M \talloblong M \;\cong_\lambda\; M \text{ for } M \text{ closed} \end{array}$$

Fig. 4. Some properties of *amb*

Theorem 3 (Soundness). *For any M, N in $\lambda^{\talloblong}$ and name p, $[\![M]\!]_p \cong_\pi [\![N]\!]_p$ implies $M \cong_\lambda N$.*

5 Deriving characteristic properties of *amb*

Figure 4 presents a set of characteristic laws of *amb* we have been able to establish. The proofs of these results are all based on the same method: we compute the $A\pi$ encoding of the two $\lambda^{\talloblong}$-terms being compared, construct a (weak or coupled) bisimulation to show (possibly using up-to techniques and algebraic laws for $A\pi$) that these processes are related by $\cong_\pi$, and conclude using Theorem 3.

We give an illustration of this method for the bottom avoidance property $M \talloblong \Omega \cong_\lambda M$, one of the key fairness properties of *amb*. We first need a technical result:

Lemma 1. *For any term M of $\lambda^{\talloblong}$ and name p, $[\![M]\!]_p \approx \boldsymbol{\nu} q \;([\![M]\!]_q \mid q \rightarrowtriangle p)$.*

This is now how we show that for any term M, $M \talloblong \Omega \cong_\lambda M$:

$$\begin{aligned} [\![M \talloblong \Omega]\!]_p &\overset{\mathsf{def}}{=} \boldsymbol{\nu} q \;\; ([\![M]\!]_q \mid [\![\Omega]\!]_q \mid q \rightarrowtriangle p) \\ &\approx \boldsymbol{\nu} q \;\; ([\![M]\!]_q \mid q \rightarrowtriangle p) && \text{because } [\![\Omega]\!]_q \approx \mathbf{0} \\ &\approx [\![M]\!]_p && \text{using Lemma 1} \end{aligned}$$

Along these lines, we prove associativity and commutativity of *amb*, and that $\cong_\lambda$ validates the β rule. We also establish associativity of internal choice ($\oplus$): it has to be stressed that for this proof, we are compelled to reason with $\leftrightarrows$ (and not $\approx$), because of the presence of 'partially committed states' in the execution of choices.

We have also established equations like $I \cong_\lambda \mathsf{Fix}(\lambda x.\ (I \oplus x))$, that holds because we consider only strong divergences (this law is not valid in the setting of [Mor98,LM99]).

Derived techniques. We can also use the π-calculus encoding to derive proof techniques similar to those used in the literature to proof the laws of $\lambda^{\talloblong}$ [Mor98,LM99]. For instance, below we derive a technique that is similar to the *"Kleene equivalence"* technique.

Definition 10 ($\asymp$). *For two $\lambda^{\talloblong}$ terms M and N, $M \asymp N$ iff*
(i) if $M \hookrightarrow^+ V$, there exists V' s.t. $N \hookrightarrow^+ V'$ and, for any p, $[\![V]\!]_p \leftrightarrows [\![V']\!]_p$;
(ii) $M \upharpoonright$ iff $N \upharpoonright$.

Proposition 3 (Soundness of $\asymp$). $\asymp \subseteq \cong_\lambda$.

Aside the use of π-calculus, the main difference with "Kleene equivalence" is that, in clause (i), the latter uses syntactic equality to compare V and V', while we can rely on behavioural

$$
\begin{array}{ll}
[\![\lambda x.M]\!]_p & \overset{\mathsf{def}}{=} \boldsymbol{\nu} l\ \ (\bar{p}\langle l\rangle \mid\, !l(x,q).[\![M]\!]_q\,) \\
[\![\mathsf{let}\ x = M\ \mathsf{in}\ N]\!]_p & \overset{\mathsf{def}}{=} \boldsymbol{\nu} q\ \ ([\![M]\!]_q \mid q(v).([\![N]\!]_p \mid\, !x(r).\bar{r}\langle v\rangle))
\end{array}
$$

Fig. 5. π-calculus encoding of $\lambda^{[\!]}$ with local call-by-value

equivalences (since $\approx \subseteq \leftrightarrows$, we can also use $\approx$ to compare $[\![V]\!]_p$ and $[\![V']\!]_p$ when treating clause (i) above). As an example of consequence of this difference, Proposition 3 allows us to show that $\lambda x.\,(x \mathbin{[\!]} \Omega) \cong_\lambda I$, which cannot be proved using the "Kleene equivalence" technique.

Full abstraction. As expected, our method is not fully-abstract with respect to $\cong_\lambda$ (because we use bisimilarity in the π-calculus whereas $\cong_\lambda$ is purely contextual; this situation is standard for encodings of λ-calculi into the π-calculus). We can however derive a *partial full-abstraction* result (partial in the sense that we only compare pure λ-terms), for the 'open' version of applicative bisimilarity. We let $\cong_\lambda^{\mathsf{op}}$ stand for the applicative bisimilarity (see Definition 9) in which two abstractions $\lambda x.\,M$ and $\lambda x.\,N$ are equated by comparing their bodies M and N (we omit the precise definition, see [SW01]).

Theorem 4 (Partial full abstraction). *Let M, N be two* pure *λ-terms, and p a name. Then*

$$[\![M]\!]_p \approx [\![N]\!]_p \qquad \textit{iff} \qquad M \cong_\lambda^{op} N\,.$$

It can be noted that for the λ-calculus extended with internal choice, the problem of full abstraction on the whole calculus (whether the π-calculus encoding is fully abstract wrt open applicative bisimilarity) is still open. The same question in $\lambda^{[\!]}$ seems at least as difficult.

6 Local call by value

An important enrichment of $\lambda^{[\!]}$ is that with the familiar $\mathsf{let}\ldots\mathsf{in}$ construction, that introduces a form of *local call by value* in the language. The corresponding additional reduction rule is:

$$\textsc{Let}\ \frac{M \hookrightarrow V}{\mathsf{let}\ x = M\ \mathsf{in}\ N \hookrightarrow N[V/x]}$$

The encoding of the resulting calculus is obtained by a modification of the encoding presented above, as shown on Figure 5 (clauses that are left unchanged are not mentioned). The translation of a $\mathsf{let}\ldots\mathsf{in}$ construct consists in the evaluation of the locally declared term, *followed by* the evaluation of the term after the "in" in which the bound variable is replaced by the computed value. We also add *persistence*, using replication, in the encoding of abstractions, since in presence of $\mathsf{let}\ldots\mathsf{in}$, several copies of a function may be triggered along a computation.

The results presented in previous sections also holds on $\lambda^{[\!]}$ with let. For instance, soundness becomes:

Theorem 5. *For any terms M, N of $\lambda^{[\!]}$ enriched with local call-by-value, and for any name p, $[\![M]\!]_p \cong_\pi [\![N]\!]_p$ implies $M \cong_\lambda N$.*

Again, using simple bisimulation reasoning, we have derived these example laws for $\lambda^{\|}$ with let:

$$\mathsf{let}\ x = V\ \mathsf{in}\ M \ \cong_\lambda\ (\lambda x.\, M)\, V \qquad \mathsf{let}\ x = M\ \mathsf{in}\ x \mathbin{[]} x \ \cong_\lambda\ M \ \text{ for } M \text{ closed}$$
$$\mathsf{let}\ x = \Omega\ \mathsf{in}\ M \ \cong_\lambda\ \Omega \qquad \mathsf{let}\ x = M\ \mathsf{in}\ N \ \cong_\lambda\ N \ \text{ if } M \Downarrow \text{ and } x \notin \mathsf{fn}(N)\,.$$

7 Further results and remarks

In the paper we have distinguished strong and weak divergences, and showed that only strong divergences should count in order to obtain a semantics to $\lambda^{\|}$ using the π-calculus. We think that both resulting semantics – the one where both strong and weak divergences count, and the one where only strong divergences count – are interesting. However, one may argue that in languages with operators like *amb*, a general fairness requirement that a computation should not "always miss a reachable value" – obtained by counting only the strong divergences – appears more reasonable (for instance, a computation starting from the term T in Section 1 should not always miss the value I).

Parallel operators that have similarities with *amb* and that have been studied in λ-calculi are those in [DCdP98] and [Bou94]. These are "parallel", rather than "choice" operators, and do not give rise to the issues of fairness that *amb* does. Indeed, semantically, these operators are much simpler than *amb*. (The encoding into the π-calculus is straightforward, see [SW01].)

In the full paper [CHS03] we discuss a few variants of the π-calculus encoding of Section 4. For instance, we discuss why certain simplifications of the encoding would not allow us to obtain some of the results in Sections 4-6 .

Some of the laws we have established express *amb*'s fairness in $A\pi$, and in our setting are derived exploiting bisimulation. It would be interesting to go further in this direction in order to gain a better understanding of the fairness brought by bisimulation. A way to do this is to study π-calculus semantics of other fair operators, like e.g. *fair merge*, which is more expressive than *amb* [PS88,FK02]. This operator computes the merge of two (finite or infinite) lists in a fair fashion, also in the case when the lists contain divergences. We have adapted an argument of [PS88] and proved that it is impossible to represent fair merge into the π-calculus at an operational level; see the full paper [CHS03]. An interesting question is the definability of fair merge *modulo bisimulation*, i.e. at a behavioural level.

Acknowledgements. We would like to thank Mariangiola Dezani for useful discussions.

References

[BL00] G. Boudol and C. Laneve. Lambda-calculus, multiplicities and the pi-calculus. In *Proof, Language, and Interaction, Essays in Honour of Robin Milner*. MIT Press, 2000.

[Bou94] G. Boudol. Lambda-calculi for (strict) parallel functions. In *Information and Computation*, volume 108, pages 51–127, 1994.

[CHS03] A. Carayol, D. Hirschkoff, and D. Sangiorgi. A π-calculus view on McCarthy's amb. full version – available at `http://www.irisa.fr/galion/acarayol/RR.ps`, 2003.

[DCdP98] M. Dezani-Ciancaglini, U. de'Liguoro, and A. Piperno. A filter model for concurrent λ-calculus. *SIAM J. Comput.*, 27(5):1376–1419, 1998.

[FK02] M. Fernandez and L. Khalil. Interaction nets with McCarthy's amb. To appear in proc. EXPRESS'02, ENTCS, 2002.

[Hen82] P. Henderson. Purely functional operating systems. In *Functional Programming and its Applications*, pages 177–192. Cambridge Univ. Press, 1982.

[HO90] J. Hughes and J. O'Donnell. Expressing and reasoning about non-deterministic functional programs. In *Proc. Glasgow 1989 Workshop on Functional Programming*, Workshops in Computing. Springer Verlag, 1990.

[How96] D. J. Howe. Proving congruence of bisimulation in functional programming languages. *Information and Computation*, 124(2):103–112, 1996.

[Las98] S. Lassen. *Relational reasoning about functions and nondeterminism.* PhD thesis, Aarhus University, 1998.

[LM99] S. Lassen and A. Moran. Unique fixed point induction for McCarthy's Amb. In *Proc. of MFCS'99*, volume 1672 of *LNCS*, pages 198–208. Springer Verlag, 1999.

[McC63] J. McCarthy. A basis for a mathematical theory of computation. In *Computer Programming and Formal Systems*, pages 33–70. North-Holland, 1963.

[Mer00] M. Merro. *Locality in the pi-calculus and applications to distributed objects.* PhD thesis, Ecole des Mines de Paris, 2000.

[Mil90] R. Milner. Functions as processes. Technical Report 1154, INRIA Sophia Antipolis, 1990.

[Mor98] A. Moran. *Call-by-name, Call-by-need, and McCarthy's Amb.* PhD thesis, Chalmers Univ. of Technology and Univ. of Gothenburg, 1998.

[NC95] V. Natarajan and R. Cleaveland. Divergence and fair testing. In *Proc. ICALP 95*, volume 944 of *LNCS*, pages 648–659. Springer Verlag, 1995.

[NP96] U. Nestmann and B. Pierce. Decoding choice encodings. In *International Conference on Concurrency Theory*, pages 179–194, 1996.

[Pit01] C. Pitcher. *Functional programming and erratic non-determinism.* PhD thesis, Oxford University, 2001.

[PS88] P. Panangaden and V. Shanbhogue. McCarthy's amb cannot implement fair merge. In *Proc. of FST-TCS*, volume 338 of *LNCS*, pages 348–363. Springer Verlag, 1988.

[San92] D. Sangiorgi. The lazy lambda calculus in a concurrency scenario. In *Proc. 7th LICS Conf.*, pages 102–109. IEEE Computer Society Press, 1992.

[San00] D. Sangiorgi. Lazy functions and mobile processes. In *Proof, Language, and Interaction, Essays in Honour of Robin Milner.* MIT Press, 2000.

[SW01] D. Sangiorgi and D. Walker. *The π-calculus: a Theory of Mobile Processes.* Cambridge University Press, 2001.

[Tur90] D. Turner. An approach to functional operating systems. In *Research Topics in Functional Programming.* Addison Wesley, 1990.