Towards action-refinement in process algebras

L. Aceto
M. Hennessy
Computer Science
University of Sussex
Falmer, Brighton BN1 9QH, England

Abstract

We present a simple process algebra which supports a form of refinement of an action by a process and address the question of an appropriate equivalence relation for it. The main result of the paper is that an adequate equivalence can be defined in a very intuitive manner and moreover can be axiomatized in much the same way as the standard behavioural equivalences.

1 Motivations

Regular expressions augmented by a shuffle operator, $\|$, have been used extensively in the theory of concurrency. This language of extended regular expressions consists of all the words which can be constructed from a set of generators using the symbols ;, +, $\|$ and $\star$. Intuitively, the generators may be looked upon as variables, as in formal language theory, or as uninterpreted actions, as in the theory of concurrency; the symbol ; represents sequential composition, + represents choice, $\|$ represents concurrency and $\star$ iteration.

We wish to re-examine the formal semantic basis for this language. Specifically, we suggest a new semantic interpretation for the language, justify it from a behavioural point of view and show that it can be finitely axiomatized. For simplicity the Kleene closure operator $\star$ is omitted. However, part of our justification involves the introduction of a new operator. In formal language terms this operator stands for the homomorphic substitution of languages for variables, whereas from the perspective of process algebras it represents the refinement of an action by a process. Indeed, much of the novelty of our approach is in the formal treatment of this operator and the investigation of the consequences of introducing it into the language.

One simple semantic theory may be based on interpreting each expression as a language of strings over the generators in the same way

CH2753-2/89/0000/0138$01.00 © 1989 IEEE

as regular expressions; a generator a is interpreted as the language $\{a\}$, ; as language concatenation, $\|$ as the shuffle of languages, $+$ as set-theoretic union and the new operator, $p[a \rightsquigarrow q]$, as language substitution.

It is well-known that this is inadequate as a semantic basis for concurrent processes. There are two problems. The first is that the interpretation of $\|$ does not properly reflect our intuitive ideas about processes running concurrently, [Pr86], [Gi84]. The second is that set-theoretic union is not adequate as an interpretation of non-determinism, [Mil80]. In [Pr86], [Gr81] and many other publications the first problem is addressed by replacing strings with partial strings (or, more formally, pomsets) but terms are still interpreted as *sets* of partial strings. In [Mil80], [HM85], on the other hand, $\|$ is interpreted as shuffle but non-determinism is handled more subtly; formally, terms not involving refinement are interpreted as equivalence classes of terms, where the equivalence is generated by an operational equivalence called *observational equivalence.*

In this paper we propose to combine both of these approaches thereby treating both non-determinism and $\|$ correctly. We justify our approach by showing that it is consistent with the idea that the semantics should only identify terms if there is no observable difference among their behaviours. We also show that the theory we propose is as elegant and as manageable as the more traditional ones by proving that it can be finitely axiomatized.

Finally this theory also supports the refinement of an action by a process, a concept which does not appear in other process algebras.

All the proofs are omitted from this presentation and can be found in the full version of the paper, [AH88].

2 The Language

The language whose semantic properties will be investigated in this paper is a *Process Algebra* in the style of **CCS** [Mil80], **CSP** [Ho85] and **ACP** [BK85], equipped with a new combinator for refining an action by a process.

Let $\mathbf{A}$ be an uninterpreted set of *actions*, ranged over by a, $b\ldots$. The set $\mathcal{EL}$ of processes over the set of generators $\mathbf{A}$ is defined by the following syntax:

$$p ::= nil \mid a \mid p + p \mid p;p \mid p|p \mid p \between p \mid p[a \rightsquigarrow p],$$

and will be ranged over by p, q, $p'\ldots$. We will use $\mathcal{BL}$ to denote the subset of $\mathcal{EL}$-processes not involving occurrences of the refinement operator $p[a \rightsquigarrow q]$.

The intuition captured by most of the operators of the calculus has already been indicated in the previous section, both in terms of process algebras and formal language theory. Following Milner [Mil80], we use the constant symbol *nil* to denote *inaction*, the process that cannot do anything; the auxiliary operator $\between$, read *left-merge*, has been introduced by Hennessy in work leading to [H88] and is

an essential technical tool to give a finite axiomatization of our semantic equivalence. We refer to [CH87] and [H88] for the discussion of the properties of $\between$. The new operator, $p[a \rightsquigarrow q]$, allows the refinement of action a by process q in the behaviour of process p. As already pointed out, this operator bears resemblance with the formal language notion of *language homomorphism* and with the notion of *pomset homomorphism* introduced in [Gi84], [Pr86].

The operational behaviour of $\mathcal{EL}$-processes can be defined by giving a standard operational semantics for $\mathcal{BL}$ following Plotkin's SOS, [Pl81], and by interpreting $p[a \rightsquigarrow q]$ as indicating the syntactical substitution of process q for each occurrence of action a in p. With this interpretation we reduce each process $p \in \mathcal{EL}$ to a process $\mathbf{red}(p) \in \mathcal{BL}$ and p's operational behaviour is determined by that of $\mathbf{red}(p)$ according to the operational semantics of $\mathcal{BL}$-processes. In the full version of the paper, [AH88], we have given an explicit operational semantics for $\mathcal{EL}$ and we have shown that the associated semantic equivalence for $\mathcal{EL}$ agrees with the one presented here.

For each $a \in \mathbf{A}$, $\xrightarrow{a}$ will denote the least binary relation on $\mathcal{BL}$ which satisfies the axiom and rules in Figure 1. Several interpretations of these relations are possible. A standard one, [Mil80], is the following:

$p \xrightarrow{a} p'$ iff p may perform action a and become p' in doing so.

The rules defining $\xrightarrow{a}$ use a termination predicate $\surd_{\mathcal{BL}}$ over terms which may easily be defined by structural induction over terms.

With the above definition we have given $(\mathcal{BL}, \mathbf{A}, \{\xrightarrow{a} \mid a \in \mathbf{A}\})$ the structure of a *Labelled Transition System* (LTS), [Kel76]. We can now define a standard observational equivalence on $\mathcal{BL}$ using the (strong) bisimulation technique introduced by Park, [Pa81]. A relation $\mathcal{R} \subseteq \mathcal{BL}^2$ is a *bisimulation* if it is symmetric and satisfies the following clause for each $(p, q) \in \mathcal{R}$, $a \in \mathbf{A}$:

$p \xrightarrow{a} p' \Rightarrow \exists q'$ such that $q \xrightarrow{a} q'$ and $(p', q') \in \mathcal{R}$.

Following standard lines it is possible to show that there exists a maximum such relation, which we will denote by $\sim$, and that $\sim = \bigcup\{\mathcal{R} \mid \mathcal{R} \text{ is a bisimulation}\}$. As argued by several authors, e. g. [Mil80], [Ab87], $\sim$ is a natural notion of equivalence on $\mathcal{BL}$ and its properties are investigated in depth in several papers in the literature, e. g. [HM85], [Mil88].

The equivalence $\sim$ can be inherited by $\mathcal{EL}$ via $\mathbf{red}(\cdot)$ as follows

$$\forall p, q \in \mathcal{EL},\ p \sim q \Leftrightarrow \mathbf{red}(p) \sim \mathbf{red}(q).$$

In this way every equivalence on $\mathcal{BL}$ can be conservatively extended to an equivalence on $\mathcal{EL}$. However, it turns out that $\sim$ is not an adequate semantic equivalence for $\mathcal{EL}$. In fact, $\sim$ is not a congruence with respect to the refinement operator, $[a \rightsquigarrow q]$. For example, it can be easily seen that $a|b \sim (a;b) + (b;a)$ but $(a|b)[a \rightsquigarrow c;d] \not\sim ((a;b)+(b;a))[a \rightsquigarrow c;d]$. This means that $\sim$ would not support compositional proof techniques for $\mathcal{EL}$. However, we have a standard way of associating an

1. $a \xrightarrow{a} nil$

2. $p \xrightarrow{a} p' \Rightarrow p + q \xrightarrow{a} p',\ q + p \xrightarrow{a} p'$

3. $p \xrightarrow{a} p' \Rightarrow p;q \xrightarrow{a} p';q$

4. $p\surd_{\mathcal{BL}},\ q \xrightarrow{a} q' \Rightarrow p;q \xrightarrow{a} q'$

5. $p \xrightarrow{a} p' \Rightarrow p|q \xrightarrow{a} p'|q,\ q|p \xrightarrow{a} q|p'$

6. $p \xrightarrow{a} p' \Rightarrow p\not|q \xrightarrow{a} p'|q$

Figure 1: Axiom and rules for $\xrightarrow{a}$

$\mathcal{EL}$-congruence with $\sim$. As shown by Milner [Mil80], it is sufficient to close $\sim$ with respect to all $\mathcal{EL}$-contexts. The resulting $\mathcal{EL}$-congruence, denoted by $\sim^c$, is known to be the largest congruence contained in $\sim$. We claim that $\sim^c$ is an adequate semantic equivalence for $\mathcal{EL}$. The basic relation $\sim$ can be viewed as arising by virtue of observing or testing processes, see [Ab87], and $\sim^c$ arises by applying these tests to processes in arbitrary contexts. However, the definition of $\sim^c$ given above does not shed light on its behavioural significance. Our main results, sketched in what follows, are that $\sim^c$ has a simple behavioural characterization and that it is mathematically tractable. In fact we show that it can be finitely axiomatized.

3 Characterizing $\sim^c$

As pointed out in the previous section, the view of processes enforced by interleaving-style equivalences is not adequate if we want to describe the behaviour of concurrent processes to allow for a refinement of the actions they perform. Essentially this depends on the fact that an interleaving view of the behaviour of concurrent processes strongly depends on what are regarded to be the atomic actions processes may perform (this point has been stressed clearly by Pratt in [Pr86]). As we allow for an operation which changes the level of atomicity of actions without enforcing any mutual exclusion policy, we can no longer consider actions as atomic events. A minimal consequence is that actions lose their "all-or-nothing" character and have distinct beginnings and endings. This is exactly the intuition underlying *timed-equivalence*, [H88]. This is a version of bisimulation equivalence based on actions which are not necessarily instantaneous. We refer the reader to [H88] for a comprehensive description of the resulting semantic theory.

3.1 Timed-equivalence

We will assume that beginnings and terminations of actions are distinct events which can be observed. For each $a \in \mathbf{A}$ we will use $S(a)$ and $F(a)$ to denote the beginning and termination of an a-action, respectively. We will view $S(a)$, $F(a)$ as a new class of events and define the operational semantics for $\mathcal{BL}$ in terms of next-event relations $\stackrel{S(a)}{\Longrightarrow}$, $\stackrel{F(a)}{\Longrightarrow}$.

Definition 3.1 $\mathbf{E} =_{def} \{S(a) \mid a \in \mathbf{A}\} \cup \{F(a) \mid a \in \mathbf{A}\}$ *will be called the set of* events *and will be ranged over by* e.

As pointed out in [H88], the language for processes is not sufficiently expressive to describe a possible state a process may reach by executing the beginning of an action. To overcome this problem we introduce into the language a new symbol $S(a)$ for each $a \in \mathbf{A}$. $S(a)$ will denote the state in which the action a is being executed but is not terminated yet.

Definition 3.2 *Let* $\mathcal{S}$, *the set of* process states, *be the least set which satisfies:*

- $p \in \mathcal{BL} \Rightarrow p \in \mathcal{S}$,
- $a \in \mathbf{A} \Rightarrow S(a) \in \mathcal{S}$,
- $s \in \mathcal{S}$, $p \in \mathcal{BL} \Rightarrow s;p \in \mathcal{S}$,
- s_1, $s_2 \in \mathcal{S} \Rightarrow s_1|s_2 \in \mathcal{S}$.

$\mathcal{S}$ *will be ranged over by* $s, s_1, s' \ldots$.

The operational semantics for process states will be defined using standard lines; the termination predicate $\surd_{\mathcal{BL}}$ can be easily extended to a predicate over $\mathcal{S}$, which we denote by $\surd$, as follows:

Definition 3.3 *For all* $s \in \mathcal{S}$, $s\surd \Leftrightarrow s \in \mathcal{BL}$ *and* $s\surd_{\mathcal{BL}}$.

For each $e \in \mathbf{E}$, the next-event relation $\stackrel{e}{\Longrightarrow}$ is defined as the least binary relation on $\mathcal{S}$ which satisfies the axioms and rules in Figure 2. Rules 1 and 2 of Figure 2 are the new rules which make explicit our view of processes. They state that an atomic process a may perform the beginning of the action a and enter state $S(a)$. Moreover, when in state $S(a)$, it can only perform the termination of the action it has started.

A standard behavioural equivalence may now be defined using the relations $\stackrel{e}{\Longrightarrow}$ and the notion of bisimulation. The resulting equivalence relation will be denoted by $\sim_t$ and will be called *timed (observational) equivalence*. The following proposition states that $\sim_t$ is contained in $\sim$ over the language $\mathcal{BL}$.

Proposition 3.1 $\forall p, q \in \mathcal{BL}$, $p \sim_t q \Rightarrow p \sim q$.

The converse implication does not hold as shown by the following example.

Example 3.1 *Consider* $p = (a;b) + (b;a)$ *and* $q = a|b$. *Then* $p \sim q$ *but* $p \not\sim_t q$.

In fact, $p \stackrel{S(a)}{\Longrightarrow} S(a);b$ *which is a state in which* p *can only perform the termination of the action* a. *No such state can be reached by* q *via* $S(a)$. *In fact,* $q \stackrel{S(a)}{\Longrightarrow} s$ *implies* $s = S(a)|b$ *and* s *may perform the start of action* b.

The equivalence $\sim_t$ can be inherited by $\mathcal{EL}$ via $\mathbf{red}(\cdot)$ as we did with $\sim$. It can be shown that $\sim_t$ is a $\mathcal{BL}$-congruence. We can now

1. $a \stackrel{S(a)}{\Longrightarrow} S(a)$

2. $S(a) \stackrel{F(a)}{\Longrightarrow} nil$

3. $p \stackrel{\epsilon}{\Longrightarrow} s \Rightarrow p+q \stackrel{\epsilon}{\Longrightarrow} s,\ q+p \stackrel{\epsilon}{\Longrightarrow} s$

4. $s \stackrel{\epsilon}{\Longrightarrow} s' \Rightarrow s;p \stackrel{\epsilon}{\Longrightarrow} s';p$

5. $s\surd,\ p \stackrel{\epsilon}{\Longrightarrow} s' \Rightarrow s;p \stackrel{\epsilon}{\Longrightarrow} s'$

6. $s_1 \stackrel{\epsilon}{\Longrightarrow} s_1' \Rightarrow s_1|s_2 \stackrel{\epsilon}{\Longrightarrow} s_1'|s_2, s_2|s_1 \stackrel{\epsilon}{\Longrightarrow} s_2|s_1'$

Figure 2: Axioms and rules for $\stackrel{\epsilon}{\Longrightarrow}$

state our first main result, namely that $\sim_t$ can be finitely axiomatized for the language $\mathcal{B}\mathcal{L}$. Let $=_E$ be the least $\mathcal{B}\mathcal{L}$-congruence which satisfies the axioms in Figure 3. We can show the following result:

Theorem 3.1 $\forall p, q \in \mathcal{B}\mathcal{L},\ p =_E q \Leftrightarrow p \sim_t q.$

This theorem shows that $\sim_t$ is as mathematically tractable as the traditional interleaving-based equivalences proposed in the literature, [HM85], [DH84].

3.2 $\sim_t$ characterizes $\sim^c$

This section is devoted to stating our main characterization result and to a sketch of its proof. The main theorem states that $\sim^c$ coincides with $\sim_t$ for the language $\mathcal{E}\mathcal{L}$.

Theorem 3.2 *For each* $p, q \in \mathcal{E}\mathcal{L},\ p \sim_t q \Leftrightarrow p \sim^c q.$

This result shows that the congruence relation $\sim^c$ on $\mathcal{E}\mathcal{L}$, defined by purely syntactical means in section 2, has indeed an intuitively appealing characterization. This result, combined with the equational characterization given in Theorem 3.1, shows that $\sim^c$ has many of the properties of the traditional interleaving semantic equivalences.

We end this account of our work with a description of the proof techniques used in showing Theorem 3.2. The complete proof of this result, which is unfortunately very technical and by no means trivial, can be found in the full version of this paper [AH88].

The *if* part of the theorem can be shown by proving that $p[\underline{a} \rightsquigarrow S(a);F(a)] \sim q[\underline{a} \rightsquigarrow S(a);F(a)]$ implies $p \sim_t q$, where we have used $[\underline{a} \rightsquigarrow S(a);F(a)]$ to denote the refinement of each action a occurring in p and q by $S(a);F(a)$.

The proof of the *only if* part is much more involved. We show that $\sim_t$ is an $\mathcal{E}\mathcal{L}$-congruence. As $\sim_t$ is contained in $\sim$ by Proposition 3.1 and $\sim^c$ is the largest $\mathcal{E}\mathcal{L}$-congruence contained in $\sim$, the desired result

$(A1)$ $(x+y)+z = x+(y+z)$

$(A2)$ $x+y = y+x$

$(A3)$ $x+x = x$

$(A4)$ $x+nil = x$

$(C1)$ $(x;y);z = x;(y;z)$

$(C2)$ $x;nil = x = nil;x$

$(C3)$ $(x+y);z = (x;z)+(y;z)$ if $x,y \notin \surd$

$(B1)$ $(x+y)\between z = x\between z + y\between z$

$(B2)$ $(x\between y)\between z = x\between(y|z)$

$(B3)$ $x\between nil = x$

$(B4)$ $nil\between x = nil$

$(X1)$ $x|y = x\between y + y\between x$

Figure 3: Complete set of axioms for $\sim_t$

will then follow.

The essential step in showing that $\sim_t$ is an $\mathcal{EL}$-congruence is proving that $\sim_t$ is preserved by the refinement operator, i. e.

$$p \sim_t q \wedge r \sim_t r' \Rightarrow p[a \rightsquigarrow r] \sim_t q[a \rightsquigarrow r'].$$

To prove this result we introduce a new equivalence relation on $\mathcal{S}$, denoted by $\sim_r$ and called *refine-equivalence*. Intuitively, $s_1 \sim_r s_2$ ensures that whenever an occurrence of $S(a)$ in s_1 is matched by an occurrence of $S(a)$ in s_2, their $F(a)$'s are also matched. We show that $\sim_r$ coincides with $\sim_t$ for our language.

Finally, using this intermediate notion of equivalence, we can show that

$$p \sim_r q \text{ and } r \sim_t r' \text{ implies } p[r/a] \sim_t q[r'/a].$$

4 References

[Ab87] S. Abramsky, Observation Equivalence as a Testing Equivalence, TCS 53, pp. 225-241, 1987

[AH88] L. Aceto and M. Hennessy, Towards Action-Refinement in Process Algebras, Report 3/88, April 1988, University of Sussex

[BK85] J. A. Bergstra and J. W. Klop, Algebra of Communicating Processes with Abstraction, TCS 37,1, pp. 77-121, 1985

[CH87] I. Castellani and M. Hennessy, Distributed Bisimulations, Report 5/87, July 1987, University of Sussex (to appear in JACM)

[DH84] R. De Nicola and M. Hennessy, Testing Equivalences for Processes, TCS 34,1, pp. 83-134, 1985

[Gi84] J. L. Gischer, *Partial Orders and the Axiomatic Theory of Shuffle*, Ph. D. Thesis, Stanford University, December 1984

[Gr81] J. Grabowski, On Partial Languages, Fundamenta Informaticae IV. 2, pp. 427-498, 1981

[H88] M. Hennessy, Axiomatising Finite Concurrent Processes, SIAM Journal of Computing, October 1988

[HM85] M. Hennessy and R. Milner, Algebraic Laws for Nondeterminism and Concurrency, J. ACM 32,1, pp. 137-161, 1985

[Ho85] C. A. R. Hoare, *Communicating Sequential Processes*, Prentice-Hall, 1985

[Kel76] R. Keller, Formal Verification of Parallel Programs, CACM 19,7, pp. 561-572, 1976

[Mil80] R. Milner, *A Calculus of Communicating Systems*, LNCS 92, Springer Verlag, 1980

[Mil88] R. Milner, Operational and Algebraic Semantics of Concurrent Processes, LFCS Report Series, ECS-LFCS-88-46, February 1988 (to appear as a chapter of the *Handbook of Theoretical Computer Science*)

[Pa81] D. Park, Concurrency and Automata on Infinite Sequences, LNCS 104, Springer Verlag, 1981

[Pl81] G. Plotkin, A Structural Approach to Operational Semantics, Report DAIMI FN-19, Aarhus University, 1981

[Pr86] V. Pratt, Modelling Concurrency with Partial Orders, International Journal of Parallel Programming 15, pp. 33-71, 1986