1992

J.W. de Bakker, E.P. de Vink

Bisimulation semantics for concurrency with atomicity and action refinement

Computer Science/Department of Software Technology Report CS-R9210 March

CWI is the research institute of the Stichting Mathematisch Centrum, which was founded on February 11, 1946, as a non-profit institution aiming at the promotion of mathematics, computer science, and their applications. It is sponsored by the Dutch Government through the Netherlands organization for scientific research (NWO).

BISIMULATION SEMANTICS FOR CONCURRENCY WITH ATOMICITY AND ACTION REFINEMENT

J.W. de Bakker
CWI, Postbus 4079, NL–1009 AB Amsterdam
& Vrije Universiteit

E.P. de Vink
Department of Mathematics and Computer Science, Vrije Universiteit
De Boelelaan 1081a, NL–1081 HV Amsterdam

Abstract

A comparative semantic study is made of two notions in concurrency, viz. atomicity and action refinement. Parallel composition is modeled by interleaving, and refinement is taken in the version where actions are refined by atomized statements. The bisimulation domain used in the semantic definitions is obtained as solution of a system of domain equations over complete metric spaces. Both operational and denotational models are developed, and their equivalence is established using higher-order techniques and Banach's fixed point theorem. The operational semantics for refinement is based on transition rules rather than on some form of syntactic substitution.

1985 Mathematics Subject Classification: 68Q10, 68Q55.
1991 Computing Reviews Categories: D.3.1, D.3.3, F.3.2, F.3.3.
Key Words and Phrases: Bisimulation semantics, concurrency, atomicity, action refinement, structured operational semantics, metric semantics.

1 Introduction

We present a semantic study of two key notions in concurrency semantics, viz. *atomicity* and *action refinement*. We develop operational and denotational models for two languages incorporating these notions as main features, and we establish equivalence of the two models for both languages.

Both languages are built from elementary actions with the operators of sequential composition (';' in its general version, i.e., not just action prefixing), choice('+'), parallel composition ('$\|$'), and recursion (see Section 3 for its syntactic form). We adopt the interleaving view of concurrency. E.g., $a \,\|\, b$ and $(a\,;b)+(b\,;a)$ will obtain the same semantics. In addition to these basic operators, the first language —which we baptize $\mathcal{L}_{at}$— has an atomization operator, denoted by $[\cdot]$. For any s, $[s]$ is its atomized version. Characteristic for this is that execution of $[s]$ proceeds in an atomic way, in the sense that no interleaving is possible (from some parallel component) during the execution of s. E.g., $[a\,;b] \,\|\, c$ has the same meaning as $(a\,;b)\,;c+c\,;(a\,;b)$ (note the absence of the summand $a\,;c\,;b$). The second language, $\mathcal{L}_{atr}$, has the same notions as $\mathcal{L}_{at}$, to which a syntactic construct for action refinement has been added: The construct $s{<}a{\rightsquigarrow}s'{>}$ expresses that occurrences of the elementary action a in s are *refined* by s'. We adopt here the *atomic* variety of refinement: During the execution of s we replace execution of (each occurrence of) an atomic a by the execution of the atomized s', i.e., by execution of $[s']$.

Report CS-R9210
ISSN 0169-118X
CWI
P.O. Box 4079, 1009 AB Amsterdam, The Netherlands

The study of atomicity and, especially, of action refinement has received a great deal of attention in recent years. For action refinement in the framework of process algebra or process description languages one may refer, e.g., to three recent Ph.D. theses ([Ace90, Gla90, Gor91]) and the many papers cited there. For refinement in the setting of Petri nets we mention the recent survey [BGV91]. Since [Pra86] or [CMP87], one has realized that the interleaving model for concurrency is incompatible with action refinement (briefly, consider $(a \,\|\, b) < a \rightsquigarrow a_1 \,;\, a_2 >$ versus $((a\,;b) + (b\,;a)) < a \rightsquigarrow a_1 \,;\, a_2 >$) *unless* the atomic view of refinement is adopted (in the example, unless $a_1 \,;\, a_2$ is atomized). Accordingly, a major part of the literature deals with refinement in the setting of true concurrency.

In exploring the atomic action refinement approach the present paper does not so much want to take a position in the debate as to which notion of refinement is to be preferred. Rather, we see our work as a case study in establishing that the general semantics methodology we have developed over the years —more about this in a moment— is applicable as well to these interesting concepts. Since we have already studied and applied notions and properties having to do with atomicity in earlier work ([BK90, B91]), it seemed natural to develop another application of the relevant machinery by connecting it with the refinement operator.

We now devote a few words to the general methodology just mentioned. Characteristic for it is the use of *complete metric spaces* (cms's) as underlying semantic domains. Operational semantics ($\mathcal{O}$) maps language constructs to elements from such a cms by employing transition systems in Plotkin's SOS style. Denotational semantics ($\mathcal{D}$) yields meanings (again in a cms) by compositional definitions. Some further characteristics of the methodology are

- the domains employed are usually obtained as solutions of (systems of) domain equations ([BZ82, AR89])
- the continuous functions from cpo-based semantics are replaced by non-distance increasing or *contracting* functions —the latter being especially useful since they have unique fixed points (by Banach's theorem)
- a variety of models in the hierarchy ranging from linear time or trace models to branching time or bisimulation models have been studied (e.g., [BBKM84, Rut89])
- a powerful proof method to relate $\mathcal{O}$ and $\mathcal{D}$ in a variety of settings has been proposed ([KR90]), and some full abstractness results have been obtained (e.g. [HBR90])
- a large number of applications to language families of the parallel logic programming and parallel object-oriented programming style have been developed (e.g., [BK90, B91, ABKR89, Rut90a])
- a beginning has been made with applying the techniques to models embodying some form of true concurrency ([BW90, BW91]).

The model we develop below for $\mathcal{L}_{at}$ and $\mathcal{L}_{atr}$ belongs to the category of branching time or bisimulation models. The 'branching time' terminology is explained by the fact that the mathematical entities (from the cms to be defined later) may be viewed as labeled tree-like objects (rather than as elements with a linear structure, such as (sets of) words). Bisimulation semantics is the familiar term for models derived from synchronization trees (e.g. [Mil80]) or process graphs (e.g. [BW90]) by identifying *bisimilar* objects. Our model —given as solution of a set of two domain equations— defines a set of labeled tree-like objects where the labels are *words* rather than just symbols. Atoms in the language return as labels in these objects, and an atom such as $[a\,;b]$ induces an edge with a label ab. Though we call our model 'bisimulation semantics' —to

adhere to generally accepted terminology— this notion does not, in fact, appear explicitly in our definitions, since the intended identifications already result from the general metric machinery. One further comment: for simplicity, our languages do not include some form of communication —or another construct which might induce deadlock. Hence, our motivation to develop the bisimulation model is not so much that this is demanded by the need to distinguish statements with different deadlock behaviour. Rather, we develop this model to illustrate the convenience of employing a simple kind of system of domain equations for an interesting application, suitable as well to accommodate possible subsequent extensions of the languages to include deadlock.
Having positioned our paper in the general field of concurrency semantics, let us now mention some more specific sources and points of interest. We employ (a refinement of) the customary SOS-style of semantics, with labeled transitions $s \xrightarrow{a} s'$ expressing that statement s can perform an a-step and then continue with s'. As new feature, we introduce a distinction into two cases, viz. transitions $s \xrightarrow{a}_1 s'$ and $s \xrightarrow{a}_2 s'$. The former will be used when, after performing an a-step, no interleaving (from a parallel component) is allowed, and the latter when interleaving after the a-step is indeed allowed. This idea was proposed in [B91] for a language with two versions of sequential composition, the new one being the 'commit' as familiar from parallel logic programming. In [BK90], we studied the atomization operator (albeit with somewhat different techniques), to be applied in an abstract version of the Concurrent Prolog language. The $\rightarrow_1$, $\rightarrow_2$ transitions are also used in [KK90]; [KK91] extends this investigation to deal with various kinds of deadlock behaviour in the presence of atomicity. The transition rules we introduce to handle action refinement are quite close to those described in [Gor91], chapter 5 (see also [DG91]). A difference is that the system used there has no explicit atomization operator; a further difference consists in the use, in [Gor91] and [DG91], of a supplementary stack mechanism in the transition rules. For a study of a notion related to atomization in the context of process algebra we refer to the discussion in [BK84] of 'tight regions'.
The bisimulation model we introduce in Section 3 —obtaining the cms's P, Q as solution of a pair of domain equations— was first presented in unpublished work by Joost Kok, together with a definition of the usual operators such as sequential and parallel composition in this setting (cf. also the remark in [BK90], p. 25). In the Appendix to the present paper, we provide full details justifying the well-definedness —and further properties— of these operators. Our denotational definitions rely mostly on the above mentioned general framework for cms-based semantics. The *semantic* operator for action refinement —a version of, possibly infinite, substitution— is new. The main technical achievement of our paper is the proof that $\mathcal{O} = \mathcal{D}$ for $\mathcal{L}_{atr}$. As usual, we apply the [KR90] method: define $\mathcal{O}$ as (unique) fixed point of a contracting higher-order mapping Φ, and then establish (with some difficulty, due to the case distinctions induced by the $\rightarrow_1/\rightarrow_2$ mechanism) that $\Phi(\mathcal{D}) = \mathcal{D}$.
We close this introduction with two points for further research. Firstly, the effects of adding a form of synchronous communication to $\mathcal{L}_{at}$ or $\mathcal{L}_{atr}$ should be explored. Secondly, we would like to study how to incorporate atomicity or action refinement in the domains of [BW90, BW91] modeling true concurrency.

2 Mathematical preliminaries

This section is mainly devoted to a summary of the basic facts from metric topology which we need in the sequel. Moreover, we recall how complete metric spaces —to be used later in the paper as semantic domains— may be determined as solutions of (systems of) domain equations.

2.1 Notations

We use the phrase: let $(x \in) X$ be a set such that ... to introduce a set X with variable x ranging over X such that .... With $\mathcal{P}(X)$ we denote the collection of all subsets of X, and with $\mathcal{P}_\pi(X)$ the collection of all subsets of X which have property π. The notation $f: X \to Y$ expresses that f is a function with domain X and range Y. If $f: X \to Y$ and $f(x) = x$ we call x a *fixed point* of f. If f has a unique fixed point, we denote it by $fix(f)$.

2.2 Metric spaces

Definition 2.1 A *metric space* is a pair (M, d) with M a nonempty set and d a mapping $d: M \times M \to [0, 1]$ (a metric or distance) that satisfies the following properties:

(i) $\forall x, y \in M: d(x, y) = 0 \Leftrightarrow x = y$
(ii) $\forall x, y \in M: d(x, y) = d(y, x)$
(iii) $\forall x, y, z \in M: d(x, z) \leq d(x, y) + d(y, z)$

We call (M, d) an *ultrametric* space if the following stronger version of property (iii) is satisfied:

(iii)$'$ $\forall x, y, z \in M: d(x, z) \leq \max\{d(x, y), d(y, z)\}$

Definition 2.2 Let (M, d) be a metric space, let $(x_i)_{i=0}^{\infty}$ (or $(x_i)_i$ for short) be a sequence in M.

a. We say that $(x_i)_i$ is a *Cauchy sequence* whenever we have $\forall \epsilon > 0\ \exists N \in \mathbb{N}\ \forall n, m \geq N$: $d(x_n, x_m) < \epsilon$.

b. Let $x \in M$. We say that $(x_i)_i$ *converges* to x and call x the *limit* of $(x_i)_i$ whenever we have $\forall \epsilon > 0\ \exists N \in \mathbb{N}\ \forall n \geq N$: $d(x, x_n) < \epsilon$. Such a sequence we call *convergent.* Notation: $\lim_{i \to \infty} x_i = x$.

c. The metric space (M, d) is called *complete* whenever each Cauchy sequence converges to an element of M.

Definition 2.3 Let (M_1, d_1), (M_2, d_2) be metric spaces.

a. We say that (M_1, d_1) and (M_2, d_2) are *isometric* if there exists a *bijection* $f: M_1 \to M_2$ such that $\forall x, y \in M_1$: $d_2(f(x), f(y)) = d_1(x, y)$. We then write $M_1 \simeq M_2$.

b. Let $\alpha \geq 0$. With $M_1 \to^\alpha M_2$ we denote the set of all functions f from M_1 to M_2 that satisfy the following property: $\forall x, y \in M_1$: $d_2(f(x), f(y)) \leq \alpha \cdot d_1(x, y)$. Functions in $M_1 \to^1 M_2$ we call *non-distance increasing* (*ndi*), functions in $M_1 \to^\alpha M_2$ with $0 \leq \alpha < 1$ we call *contracting of factor* α.

Remark Throughout the paper we shall, for simplicity, for each α-contracting function f, have (or assume) that $\alpha = \frac{1}{2}$.

Theorem 2.4 (*Banach's fixed point theorem*) Let (M, d) be a complete metric space (*cms*, for short) and let $f: M \to M$ be contracting. Then there exists $x \in M$ such that

(i) $f(x) = x$ (x is a fixed point of f),

(ii) $\forall y \in M: f(y) = y \Rightarrow x = y$ (x is unique),

(iii) $\forall x_0 \in M: \lim_{n\to\infty} f^n(x_0) = x$ where $f^{n+1}(x_0) = f(f^n(x_0))$, $f^0(x_0) = x_0$.

Banach's theorem thus states that a contraction f has a unique fixed point. We will denote this fixed point by $fix(f)$.

Definition 2.5 A subset X of a metric space (M, d) is called *compact* whenever each sequence in X has a subsequence that converges to an element of X.

Each compact set X is *closed* (i.e., each Cauchy sequence in X converges to an element of X). The main role of the compactness property for our purposes is based on the theorems of Kuratowski ([Kur56]) and Michael ([Mic51]). The former states that the space of compact subsets (equipped with a suitable metric) of a complete space is itself complete. The latter is useful for showing the well-definedness of certain semantic operators.

Definition 2.6 Let (M, d), (M_1, d_1), (M_2, d_2) be metric spaces.

a. We define a metric d_F on $M_1 \to M_2$ as follows: for every $f_1, f_2 \in M_1 \to M_2$,

$$d_F(f_1, f_2) = \sup\{d_2(f_1(x), f_2(x)) \mid x \in M_1\}.$$

For $\alpha \geq 0$ the set $M_1 \to^\alpha M_2$ is a subset of $M_1 \to M_2$, and the metric on $M_1 \to^\alpha M_2$ can be obtained by taking the restriction of the corresponding d_F.

b. With $M_1 \dot{\cup} M_2$ we denote the disjoint union of M_1 and M_2, which can be defined as $\{1\} \times M_1 \cup \{2\} \times M_2$. We define a metric on $M_1 \dot{\cup} M_2$ as follows: for every $x, y \in M_1 \dot{\cup} M_2$,

$$d_U(x, y) = \begin{cases} d_i(x, y) & \text{if } x, y \in \{i\} \times M_i,\ i = 1 \text{ or } 2 \\ 1 & \text{otherwise} \end{cases}$$

c. We define a metric d_P on $M_1 \times M_2$ by the following clause:

$$d_P((x_1, x_2), (y_1, y_2)) = \max\{d_1(x_1, y_1), d_2(x_2, y_2)\}.$$

d. Let $\mathcal{P}_{nc}(M) = \{X \subseteq M \mid X \text{ nonempty and compact }\}$. We define a metric d_H on $\mathcal{P}_{nc}(M)$, called the Hausdorff distance, as follows: For every $X, Y \in \mathcal{P}_{nc}(M)$,

$$d_H(X, Y) = \max\{\sup_{x\in X}\{d(x, Y)\}, \sup_{y\in Y}\{d(y, X)\}\}$$

where $d(w, Z) = \inf_{z\in Z}\{d(w, z)\}$, for every $Z \in \mathcal{P}_{nc}(M)$, $w \in M$.
In $\mathcal{P}_{co}(M) = \{X \subseteq M \mid X \text{ compact }\}$ we also have the empty set as an element. We define d_H as above but extended with the following case: If $X \neq \emptyset$ then $d_H(X, \emptyset) = d_H(\emptyset, X) = 1$. Also we put $d_H(\emptyset, \emptyset) = 0$.

e. Let $\alpha > 0$. We define $id_\alpha(M, d) = (M, \alpha \cdot d)$.

Theorem 2.7 Let (M,d), (M_1,d_1) (M_2,d_2), d_F, d_U, d_P and d_H be as in Definition 2.6, and suppose that (M,d), (M_1,d_1) (M_2,d_2) are complete. We have that

a. $(M_1 \to M_2, d_F)$, $(M_1 \to^{\alpha} M_2, d_F)$

b. $(M_1 \dot{\cup} M_2, d_U)$

c. $(M_1 \times M_2, d_P)$

d. $(\mathcal{P}_{nc}(M), d_H)$, $(\mathcal{P}_{co}(M), d_H)$

are complete metric spaces. If (M,d) and (M_i,d_i), i=1,2, are ultrametric spaces then these composed spaces are again ultrametric.

The proofs of Theorem 2.7, parts a, b, c are straightforward. Part d is more involved. It can be proved with the help of the following characterization:

Theorem 2.8 Let $(\mathcal{P}_{nc}(M), d_H)$ be as in Definition 2.6. Let $(X_i)_i$ be a Cauchy sequence in $\mathcal{P}_{nc}(M)$. We have $\lim_i X_i = \{\lim_i x_i \mid x_i \in X_i,\ (x_i)_i \text{ a Cauchy sequence in } M\}$.

The proofs of Theorem 2.7d and Theorem 2.8 are due to Kuratowski ([Kur56]), as a generalization of a similar result for *closed* subsets, see e.g., [Hah48].

The following properties of the Hausdorff distance are sometimes convenient.

Lemma 2.9

a. Define for $X, Y \in \mathcal{P}_{co}(M)$

$$\tilde{d}_H(X,Y) = \inf\{\epsilon \mid \forall x \in X \exists y \in Y : d(x,y) \leq \epsilon,\ \forall y \in Y \exists x \in X : d(y,x) \leq \epsilon\}$$

Then $\tilde{d}_H = d_H$.

b. For $X, Y \in \mathcal{P}_{nc}(M)$ it holds that

$$\forall x \in X \exists y \in Y : d(x,y) \leq d(X,Y) \text{ and } \forall y \in Y \exists x \in X : d(y,x) \leq d(X,Y)$$

c. For each index set I and $\{X_i, Y_i \mid i \in I\} \subseteq \mathcal{P}_{co}(M)$ it holds that

$$d(\cup_{i \in I} X_i, \cup_{i \in I} Y_i) \leq \sup\{d(X_i, Y_i) \mid i \in I\}$$

We conclude this subsection with the important

Theorem 2.10 (Michael) Let $X \in \mathcal{P}_{co}(\mathcal{P}_{co}(M))$. Then $\bigcup X \in \mathcal{P}_{co}(M)$.

Theorem 2.10 will be used (e.g., in Lemma A.5) in a setting where X consists of a family of compact subsets of (M,d), i.e., $X = \{A_i \mid A_i \in \mathcal{P}_{co}(M), i \in I\}$, for some index set I, and the set X is itself a compact subset of $(\mathcal{P}_{co}(M), d_H)$. The theorem then states that $\bigcup X$, i.e., $\bigcup\{A_i \mid i \in I\}$, resides in $\mathcal{P}_{co}(M)$.

Finally we state a useful lemma, which is an immediate consequence of Theorem 2.4. (See, e.g., the proof of Lemma A.3 in the appendix.)

Lemma 2.11 Let M be a cms and $f : M \to M$ be a contraction. Suppose X is a nonempty and closed subset of M such that $f(X) \subseteq X$. Then it holds that $fix(f) \in X$.

2.3 Domain equations

In [BZ82, AR89] a method has been developed to determine complete (ultra)metric spaces as solutions of domain equations of the form

$$P \simeq \mathcal{F}(P) \tag{2.1}$$

where $\mathcal{F}$ is a functor (on a category of cms's) satisfying certain conditions. Natural examples of $\mathcal{F}$ are obtained by building it in terms of the operations on metric spaces encountered in Definition 2.6. The theory may be generalized in a straightforward way to *systems* of domain equations

$$P_i \simeq \mathcal{F}_i(P_1, \ldots, P_n), \qquad n \geq 1, i = 1, \ldots, n \tag{2.2}$$

We shall restrict ourselves to the discussion of one example of (2.1). In the semantics to be presented in Section 3, we shall encounter an example of a system (with $n = 2$). Some familiarity with the example to be discussed now will enhance the subsequent understanding of that system. We shall be concerned with (P, d) —or P, for short—, solving the domain equation

$$P \simeq \mathcal{P}_{co}\Big(A \cup (A \times id_{\frac{1}{2}}(P))\Big) \tag{2.3}$$

(We use $\cup$ rather than $\bar{\cup}$ since the arguments are already disjoint.) From now on, we shall simply write $P/2$ for $id_{\frac{1}{2}}(P)$. Elements p in P will be called *processes*. The equation (2.3) assumes the discrete metric on A and, moreover, the various induced metrics (for $\cup$, $\times$, $\mathcal{P}_{co}$, $id_{\frac{1}{2}}$) as defined earlier. As a consequence, the metric d on P equals the Hausdorff metric d_H induced by the following metric $\bar{d}$ on $A \cup (A \times P/2)$:

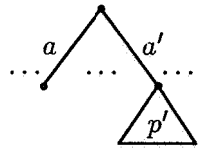
$$\bar{d}(a_1, a_2) = \begin{cases} 0 & \text{if } a_1 = a_2 \\ 1 & \text{if } a_1 \neq a_2 \end{cases}$$

$$\bar{d}(<a_1, p_1>, <a_2, p_2>) = \begin{cases} \frac{1}{2} d(p_1, p_2) & \text{if } a_1 = a_2 \\ 1 & \text{if } a_1 \neq a_2 \end{cases}$$

$$\bar{d}(a, <a', p'>) = \bar{d}(<a', p'>, a) = 1$$

Note Since the discrete metric on A is certainly an ultrametric, the general theory implies that (P, d) is a complete *ultra*metric space.

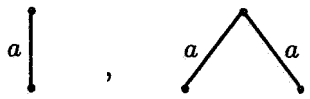
A process $p \in P$ may be viewed as a tree-like object. It consists of a (possibly empty) set of elements, each of the form $a \in A$ or $<a', p'> \in A \times P$. If p is empty, no action is possible. Otherwise, p may choose between 'steps' a or $<a', p'>$, where p' is itself a process. In a picture, a process $p(\neq \emptyset)$ may be represented as



Each such 'tree' is

- commutative

- absorptive: the successors of any node form a set rather than a multiset; e.g., we no not distinguish between processes represented by the two trees

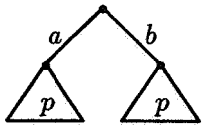


- compact: it will turn out, that this is the appropriate generalization of 'finitely branching' in the present setting

Examples of processes:

1. $\emptyset$, $\{a\}$, $\{a,b\}$, $\{<a_1,\{a_2,a_3\}>\}$, $\{<a_1,\{a_2\}>,<a_1,\{a_3\}>\}$.

2. The process determined by $p = \lim_i p_i$, p_0 arbitrary, $p_{i+1} = \{<a,p_i>,<b,p_i>\}$. This p may be depicted —in a recursive manner— as



3. Let us, informally, define the operation of sequential composition $\circ$: $P \times P \to P$, by putting: $p_1 \circ p_2$ is the process obtained by replacing, in p_1, all leaves $\{a\}$ by $\{<a,p_2>\}$. Now let p be defined as the process satisfying $p = \{a\} \cup (\{a\} \circ p)$. Since p is compact (hence closed), it must include, besides all finite elements $<a,\{<a,\ \ldots,\ \{a\}>\}>$, also the infinite element $<a,\{<a,\ \ldots>\}>$.
(Jeroen Warmerdam has shown (personal communication) that the operation of sequential composition sketched above is not well-defined if processes are only required to be closed and infinite alphabets are allowed.)

We conclude this subsection with two further remarks on processes determined by (2.3):

- ((2.3) as a system) To prepare the way for the system of equations to be discussed in Section 3, we remark that, clearly, P also may be given as the first component of the solution of

$$\begin{array}{lcl} P & \simeq & \mathcal{P}_{co}(Q) \\ Q & \simeq & A \cup (A \times P/2) \end{array} \tag{2.4}$$

- (bisimulation) Let us call two processes p_1, p_2 *bisimilar* (denoted by $p_1 \leftrightarrows p_2$) if there exists a *bisimulation* R such that $p_1 R p_2$. Here a bisimulation is a relation R on $P \times P$ satisfying: if $p_1 R p_2$ then
  - if $a \in p_1$ then $a \in p_2$
  - if $a \in p_2$ then $a \in p_1$
  - if $<a,p'> \in p_1$ then there exists p'' such that $<a,p''> \in p_2$ and $p' R p''$,
  - if $<a,p''> \in p_2$ then there exists p' such that $<a,p'> \in p_1$ and $p' R p''$.

It is not difficult to show (cf. a very similar proof in [GR89]) that, for P as in (2.3), we have: $p_1 \leftrightarrows p_2$ iff $p_1 = p_2$. (Hint: Take $\epsilon \stackrel{df}{=} \sup\{d(p_1,p_2) \mid p_1 \leftrightarrows p_2\}$, and prove $\epsilon \leq \frac{1}{2}\epsilon$, hence $\epsilon = 0$.) Thus, in P we have the attractive situation that, contrary to the general setting with synchronization trees ([Mil80]) or process graphs ([BW90]), there is no need to divide by the bisimilarity equivalence: bisimilar processes are 'automatically' identified in P.

3 Atomicity – syntax and operational semantics

3.1 Syntax for $\mathcal{L}_{at}$

We introduce a simple language $\mathcal{L}_{at}$ with concurrency and an atomization operator. Throughout the paper, we use a self-explanatory BNF-like notation for syntactic definitions. We start from the following two basic syntactic sets:

- $(a \in) A$, the (not necessarily finite) alphabet of *elementary actions*
- $(x \in) PVar$, the alphabet of *procedure variables*

Definition 3.1

a. The class $(s \in) \mathcal{L}_{at}$ of *statements* is given by

$$s ::= a \mid x \mid s_1 \,;\, s_2 \mid s_1 + s_2 \mid s_1 \parallel s_2 \mid [s]$$

b. The class $(g \in) \mathcal{L}^g_{at}$ of *guarded statements* is given by

$$g ::= a \mid g \,;\, s \mid g_1 + g_2 \mid g_1 \parallel g_2 \mid [g]$$

c. The class $(d \in) Decl_{at}$ of *declarations* consists of mappings $d\colon PVar \to \mathcal{L}^g_{at}$.

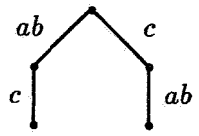
d. A *program* is a pair (d, s).

Remarks

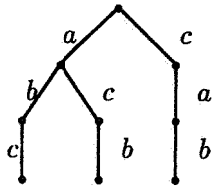
1. We do not bother about possible syntactic (and ensuing semantic) ambiguities, which may be resolved by suitable use of parentheses.
2. The guardedness (or 'Greibach') condition ensures that the execution of each procedure body (each $d(x)$, for $x \in PVar$) starts with the execution of an elementary action (rather than of a procedure variable). Technically, this condition is necessary for the well-definedness of the complexity function c, cf. Definition 3.3.

Examples Take $PVar = \{x\}$, $A = \{a, b, c, \ldots\}$, and write $x \Leftarrow g$ for $d(x) = g$. Possible programs are $(x \Leftarrow (a \,;\, x \,;\, b) + c,\ x)$ or $(x \Leftarrow [a \,;\, x] + b,\ x \parallel c)$. The construct $(x \Leftarrow x \,;\, b + a,\ x)$ is not a program, since $x \,;\, b + a \notin \mathcal{L}^g_{at}$.

The operator $[\cdot]$ turns a statement s into its atomized version $[s]$ which acts as an elementary action. In particular, it is not interruptible by actions from some parallel component: according to the (interleaving branching time) semantics to be developed presently, $[a\,;b] \parallel c$ has as its meaning (the mathematical object represented by) the tree



whereas $(a\,;b) \parallel c$ has as its meaning the tree



3.2 Operational semantics

We shall use the auxiliary symbol E (with $\mathrm{E} \notin \mathcal{L}_{at}$) to denote termination, and use $\bar{s}$ to range over $\mathcal{L}_{at} \cup \{\mathrm{E}\}$.
The operational semantics $\mathcal{O}$ will be given based on a *labeled transition system* $\mathcal{T}_{at}$. $\mathcal{T}_{at}$ determines a transition relation $\mathcal{R}_{at}$ which is given as the least relation satisfying, in the natural way, the axioms and rules of $\mathcal{T}_{at}$. The transitions in $\mathcal{T}_{at}$ are of three forms: $s \xrightarrow{a}_0 \mathrm{E}$, $s \xrightarrow{a}_1 s'$, or $s \xrightarrow{a}_2 s'$ (suppressing, for the moment, explicit mentioning of the declaration d, see later). The index i in $\rightarrow_i$, i=0, 1 or 2, plays an important role in the design of $\mathcal{T}_{at}$, in particular tuned to the handling of $[\cdot]$ (cf. [B91]). We recall that, in the basic interleaving model for a CCS-like language, a transition $s \xrightarrow{a} s'$ allows possible parallel actions to interleave with the execution of s at the point where the action a has been completed (and the resumption s' has not yet taken its own first step). More specifically, in a situation where the transitions

$$s \xrightarrow{a} s', \quad s_1 \xrightarrow{a_1} s_1', \quad s' \xrightarrow{a'} s''$$

are possible, for $s \between s_1$ ($\between$ the usual leftmerge) we have as possibilities

$$s \between s_1 \xrightarrow{a} s' \parallel s_1 \xrightarrow{a_1} s' \parallel s_1'$$

(execution of s is interleaved with an action a_1 taken by s_1) *and*

$$s \between s_1 \xrightarrow{a} s' \parallel s_1 \xrightarrow{a'} s'' \parallel s_1$$

Now the system $\mathcal{T}_{at}$ is designed in such a way that a transition $s \xrightarrow{a}_1 s'$ allows only the derivation

$$s \between s_1 \xrightarrow{a}_1 s' \between s_1 \xrightarrow{a'} s'' \between s_1$$

(no interleaving from s_1), whereas from $s \xrightarrow{a}_2 s'$ we may derive both

$$s \between s_1 \xrightarrow{a}_2 s' \parallel s_1 \xrightarrow{a_1}_2 s' \parallel s_1'$$
$$s \between s_1 \xrightarrow{a}_2 s' \parallel s_1 \xrightarrow{a'}_2 s'' \parallel s_1$$

We shall say more about $\rightarrow_i$ when we discuss how to obtain $\mathcal{O}$ from $\mathcal{T}_{at}$.

Let $\mathcal{L}_{at}^{+}$ be the extension of $\mathcal{L}_{at}$ which also includes statements of the form $s_1 \,[\!]\, s_2$ and let $\bar{\mathcal{L}}_{at}^{+} = \mathcal{L}_{at}^{+} \cup \{\mathrm{E}\}$. We introduce $\mathcal{T}_{at}$ in

Definition 3.2 The transition system $\mathcal{T}_{at}$ (and its induced relation $\mathcal{R}_{at}$) is given as follows:

a. A transition is a fourtuple $(s, <a, i>, d, \bar{s})$ in $\mathcal{L}_{at}^{+} \times (A \times \{0,1,2\}) \times Decl_{at} \times \bar{\mathcal{L}}_{at}^{+}$; we usually write it as

$$s \xrightarrow{a}_{i,d} \bar{s}$$

b. The axioms and rules of $\mathcal{T}_{at}$ are as follows

- $$a \xrightarrow{a}_{0,d} \mathrm{E} \qquad \text{(el.action)}$$
- $$\frac{g \xrightarrow{a}_{i,d} \bar{s}}{x \xrightarrow{a}_{i,d} \bar{s}} \qquad i = 0,\ 1 \text{ or } 2,\ d(x) = g \qquad \text{(recursion)}$$
- $$\frac{s \xrightarrow{a}_{i,d} \bar{s}}{s + s_1 \xrightarrow{a}_{i,d} \bar{s}} \qquad \frac{}{s_1 + s \xrightarrow{a}_{i,d} \bar{s}} \qquad i = 0,\ 1 \text{ or } 2 \qquad \text{(choice)}$$
- $$\frac{s_1 \,[\!]\, s_2 \xrightarrow{a}_{i,d} \bar{s}}{s_1 \,\|\, s_2 \xrightarrow{a}_{i,d} \bar{s}} \qquad \frac{}{s_2 \,\|\, s_1 \xrightarrow{a}_{i,d} \bar{s}} \qquad i = 0,\ 1 \text{ or } 2 \qquad \text{(merge)}$$
- $$\frac{s \xrightarrow{a}_{0,d} \mathrm{E}}{s\,;\, s_1 \xrightarrow{a}_{2,d} s_1} \qquad \frac{}{s \,[\!]\, s_1 \xrightarrow{a}_{2,d} s_1} \qquad \frac{}{[s] \xrightarrow{a}_{0,d} \mathrm{E}} \qquad \left.\begin{array}{l}\\ \\ \\ \end{array}\right\} (\rightarrow_2\text{-intro}) \qquad (\rightarrow_0\text{-premise})$$
- $$\frac{s_1 \xrightarrow{a}_{1,d} s_2}{s_1\,;\, s \xrightarrow{a}_{1,d} s_2\,;\, s} \qquad \frac{}{s_1 \,[\!]\, s \xrightarrow{a}_{1,d} s_2 \,[\!]\, s} \qquad \frac{}{[s_1] \xrightarrow{a}_{1,d} [s_2]} \qquad (\rightarrow_1\text{-premise})$$
- $$\frac{s_1 \xrightarrow{a}_{2,d} s_2}{s_1\,;\, s \xrightarrow{a}_{2,d} s_2\,;\, s} \qquad \frac{}{s_1 \,[\!]\, s \xrightarrow{a}_{2,d} s_2 \,\|\, s} \qquad \frac{}{[s_1] \xrightarrow{a}_{1,d} [s_2]} \qquad (\rightarrow_1\text{-intro}) \qquad (\rightarrow_2\text{-premise})$$

Instead of $s \xrightarrow{a}_{i,d} \bar{s} \in \mathcal{R}_{at}$ (the transition $s \xrightarrow{a}_{i,d} \bar{s}$ can be derived from $\mathcal{T}_{at}$) we always simply write $s \xrightarrow{a}_{i,d} \bar{s}$. Moreover, we usually drop the index d: Contrary to what is the case for statements, declarations remain fixed — here and in subsequent sections — throughout the various arguments, and no ambiguities are caused by not explicitly writing d.
An important property of $\mathcal{T}_{at}$ is that it is *finitely branching*. The proof of this claim (Lemma 3.4) is based on a complexity measure $c(s)$, for $s \in \mathcal{L}^+_{at}$, a notion which will also be applied frequently in the sequel:

Definition 3.3 The mapping $c: \mathcal{L}^+_{at} \to \mathrm{N}$ is defined by

- $c(a) = 1$, $c(x) = c(d(x)) + 1$
- $c(s_1 \,;\, s_2) = c(s_1) + 1$
- $c(s_1 + s_2) = c(s_1 \between s_2) = c(s_1) + c(s_2) + 1$, $c(s_1 \parallel s_2) = c(s_1) + c(s_2) + 2$
- $c([s]) = c(s) + 1$

Note that c depends as well on d. We easily see that $c(s)$ is well-defined (by induction on the syntactic complexity of first, $g \in \mathcal{L}^g_{at}$, and, next, of any $s \in \mathcal{L}^+_{at}$). Next, we may use induction on $c(s)$ in the proof of

Lemma 3.4 For each s, the set $\{(a, s') \mid s \xrightarrow{a}_i s',\ i{=}0,\ 1 \text{ or } 2\}$ is finite.
Proof Left to the reader. □

We next discuss how to obtain $\mathcal{O}(d, s)$ — the operational semantics of (d, s) — by collecting in some way the successive transitions determined by $\mathcal{T}_{at}$ for (d, s). An important decision concerns the design of the domain P in which the meanings $\mathcal{O}(d, s)$ are to be found. As indicated earlier, we want to equip P with a 'branching time' (BT) structure, a terminology referring to the tree-like structure of the entities in P. We also already mentioned that we shall apply the metric machinery (of Section 2) for the definition of P. This has as a bonus that there is no need to introduce a relation of *bisimilarity* on P, since this will turn out to coincide with the identity relation anyway (more about this at the end of Section 3). The 'trees' which will appear in our model are labeled; moreover, as labels we do not just have symbols from some alphabet A — the customary case elsewhere — but rather *words* over A (or, more precisely, a notion isomorphic to this). Each such label represents (the meaning of) an atomic action. Accordingly (repeating the examples of the introduction to the present section), we shall be able to use the tree

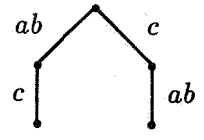


Figure 1: $[a \,;\, b] \parallel c$

for the meaning of $[a \,;\, b] \parallel c$, and the tree

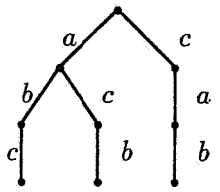


Figure 2: $(a\,;b) \parallel c$

for the meaning of $(a\,;b) \parallel c$. We proceed with the precise definition of P.

Definition 3.5 The domains P, Q are given as (the first and second component of the solution of) the system of equations

$$\begin{aligned} P &\simeq \mathcal{P}_{nc}(Q) \\ Q &\simeq A \cup (A \times Q/2) \cup (A \times P/2) \end{aligned} \tag{3.1}$$

Let us repeat the notational conventions of Section 2: A is the set of atomic actions (we do not distinguish between the 'syntactic' A and the 'semantic' A), equipped with the discrete metric; $\simeq$ denotes isometry, $\mathcal{P}_{nc}(\cdot)$ all nonempty compact subsets of $(\cdot)$, and $Q/2$, $P/2$ are short for $id_{\frac{1}{2}}(Q)$, $id_{\frac{1}{2}}(P)$. We do not repeat the conventions about the underlying metrics which are in fact part of the equations (3.1).
We see that each $p \in P$ is a compact set, the elements of which are either atomic actions ($a \in A$), or pairs of the form $<a,q>$ or $<a,p>$. There is an essential difference between (the way we shall use) $<a,q>$ and $<a,p>$ in that the former represents the situation where, after having performed a we continue with q *without* the possibility of interleaving from a parallel component at this point, whereas in $<a,p>$ we do have the possibility to interrupt after a and before continuing with p. The trees depicted in the two figures just given are denoted by the following elements in P:

(for Figure 1) $p_1 = \{<a,<b,\{c\}>>,<c,\{<a,b>\}>\}$
(for Figure 2) $p_2 = \{<a,\{<b,\{c\}>,<c,\{b\}>\}>,<c,\{<a,\{b\}>\}>\}$

In Section 4, we shall define the semantic operator $\|: P \times P \to P$ yielding the meaning of the syntactic $\|$ (we do not bother to distinguish in notation between these two $\|$) in such a way that the intended interpretation of $<a,q>$ versus $<a,p>$ is indeed satisfied.

We now connect our earlier discussion on the role of the different $\xrightarrow{a}_i$ transitions to the design of the domain P. We shall determine the operational semantics $\mathcal{O}(d,s)$ based on the transition system $\mathcal{T}_{at}$. Since, as already pointed out above, d remains fixed (s is the only interesting argument), we use instead the notation $\mathcal{O}_d(s)$, and we discuss how to determine $\mathcal{O}_d: \mathcal{L}_{at} \to P$. Three cases arise, depending on the transition steps prescribed by $\mathcal{T}_{at}$: A step $s \xrightarrow{a}_0 \mathrm{E}$ simply results in the action a as part of the final outcome. A step $s \xrightarrow{a}_1 \bar{s}$ results in a set of possible outcomes, viz. in all $<a,q>$ such that q is an element of $\mathcal{O}_d(\bar{s})$. Note that, in this way, $\xrightarrow{a}_1$ steps *have* to continue with the execution of the right-hand side $\bar{s}$; this is consistent with our earlier explanations. Thirdly, a step $s \xrightarrow{a}_2 \bar{s}$ contributes the pair $<a,\mathcal{O}_d(\bar{s})>$ to the outcome. Since this pair is of the $<a,p>$ variety, there is indeed the option of having an interleaving action (from some parallel component) after this a. Once more inspecting $\mathcal{T}_{at}$, we see that the impossibility/possibility of having interleaving actions according to $\xrightarrow{a}_1/\xrightarrow{a}_2$ steps is reflected in

the different handling of the $\lfloor\!\lfloor$-operator in rules ($\rightarrow_1$-premise) and ($\rightarrow_2$-premise), respectively: Whereas in the former we infer a transition

$$s_1 \lfloor\!\lfloor s \xrightarrow{a}_1 s_2 \lfloor\!\lfloor s$$

(thus enforcing, by the $\lfloor\!\lfloor$ on the right-hand side, that s_2 is to be executed next), in the latter we infer

$$s_1 \lfloor\!\lfloor s \xrightarrow{a}_2 s_2 \parallel s$$

thus allowing as well that, after step a, an interleaving step is performed by s.
Furthermore, the rules dealing with $[s]$ are designed in such a way that only steps $[s]\xrightarrow{a}_0 \mathrm{E}$ and $[s]\xrightarrow{a}_1 s'$ (and no $[s]\xrightarrow{a}_2 s''$) are possible. The intended meaning: no interleaving while $[s]$ is executed, is indeed achieved.
Putting the three cases together, we want that $\mathcal{O}_d(s)$ satisfies the following relationship:

$$\mathcal{O}_d(s) = \{\, a \mid s \xrightarrow{a}_0 \mathrm{E} \,\} \cup \{\, <a,q> \mid s \xrightarrow{a}_1 s',\ q \in \mathcal{O}_d(s') \,\} \cup \{\, <a, \mathcal{O}_d(s'')> \mid s \xrightarrow{a}_2 s'' \,\}$$

Clearly, this does not result in a proper definition, since $\mathcal{O}_d$ returns on the right-hand side of the definition. Therefore, we take the usual approach in such a situation: we define $\mathcal{O}_d$ as (unique) fixed point of an (higher-order) operator Φ_d, as is done in

Definition 3.6

a. Let $F \in \mathcal{L}^+_{at} \rightarrow P$, and let $\Phi_d\colon (\mathcal{L}^+_{at} \rightarrow P) \rightarrow (\mathcal{L}^+_{at} \rightarrow P)$ be given by

$$\Phi_d(F)(s) = \{\, a \mid s \xrightarrow{a}_0 \mathrm{E} \,\} \cup \{\, <a,q> \mid s \xrightarrow{a}_1 s',\ q \in F(s') \,\} \cup \{\, <a, F(s'')> \mid s \xrightarrow{a}_2 s'' \,\}$$

b. $\mathcal{O}_d = \mathit{fix}(\Phi_d)$.

Clause b of this definition is justified by

Lemma 3.7

a. $\Phi_d(F)$ is well-defined for each (F and) s.

b. Φ_d is contracting in F.

Proof Part a, in particular the compactness of $\Phi_d(F)(s)$ for each s, follows since $\mathcal{T}_{at}$ is finitely branching. Part b is clear by the definition, in particular by the $<a,\ldots>$ steps which precede application of F on the right-hand side. □

Examples

1. Since we have

$$[a\,;b] \lfloor\!\lfloor c \xrightarrow{a}_1 [b] \lfloor\!\lfloor c \xrightarrow{b}_2 c \xrightarrow{c}_0 \mathrm{E}$$
$$c \lfloor\!\lfloor [a\,;b] \xrightarrow{c}_2 [a\,;b] \xrightarrow{a}_1 [b] \xrightarrow{b}_0 \mathrm{E}$$

we obtain (for arbitrary d)

$$\mathcal{O}_d([a\,;b] \parallel c) = \{<a,<b,\{c\}>>,<c,\{<a,b>\}>\}$$

2. Abbreviating $<a, <b, \ldots, p>>$ to $<ab\cdots, p>$ (and similarly with q replacing p), we have

$$\mathcal{O}(x \Leftarrow a\,;\, x + b, [x] \parallel c) =$$

$$\{<a^n b, \{c\}> \mid n = 0, 1, \ldots\} \cup \{a^\omega\} \cup \{<c, \{a^n b \mid n = 0, 1, \ldots\} \cup \{a^\omega\}>\}$$

We conclude this section with the following

Remark We still owe the reader some comments on bisimulation in the present setting: A *bisimulation* on (P, Q) is a pair of relations (R, S) with $R \subseteq P \times P$, $S \subseteq Q \times Q$ which satisfies

1. If $p_1 \, R \, p_2$ then for all $q_1 \in p_1$ there exists $q_2 \in p_2$ such that $q_1 \, S \, q_2$, and symmetric
2. If $q_1 \, S \, q_2$ then
   - if $q_1 \in A$, then $q_1 = q_2$, and symmetric
   - if $q_1 = <a, q'>$ then there exists q'' such that $q_2 = <a, q''>$ and $q' \, S \, q''$, and symmetric
   - if $q_1 = <a, p'>$ then there exists p'' such that $q_2 = <a, p''>$ and $p' \, R \, p''$, and symmetric

We say that (p_1, q_1) and (p_2, q_2) are *bisimilar* (written as $(p_1, q_1) \leftrightarrows (p_2, q_2)$) whenever there exists a bisimulation (R, S) with $p_1 \, R \, p_2$ and $q_1 \, S \, q_2$. Similar to the result at the end of Section 2, we have $(p_1, q_1) \leftrightarrows (p_2, q_2)$ iff $(p_1, q_1) = (p_2, q_2)$.

4 Denotational semantics, $\mathcal{O} = \mathcal{D}$

The denotational semantics $\mathcal{D}(d, s)$ —with (d, s) as in Section 3— is, similarly to what we did above, expressed in terms of a mapping $\mathcal{D}_d : \mathcal{L}_{at} \to P$. Thus, $\mathcal{D}_d$ has the same domain and codomain as $\mathcal{O}_d$. Now, instead of basing the definition on $\mathcal{T}_{at}$, we provide a *compositional* definition for $\mathcal{D}_d$. We find it convenient (though not essential) to determine $\mathcal{D}_d$ as well as fixed point of a higher-order operator. As preparatory step, we define the semantic operators $\circ$, $\cup$ and $\parallel$ (and the auxiliary $\between$), used in the meanings of $s_1 \,;\, s_2$, $s_1 + s_2$, and $s_1 \parallel s_2$. We shall not be fully rigorous in these definitions. Rather, we give definitions where the semantic operators are defined in terms of themselves and we refer to the Appendix for a treatment with the appropriate (fixed points of) higher-order mappings.

Definition 4.1

a. The mappings $\circ : P \times P \to P$ and $\circ' : Q \times P \to Q$ are given by

$$\begin{aligned} p_1 \circ p_2 &= \{q \circ' p_2 \mid q \in p_1\} \\ a \circ' p &= <a, p> \\ <a, q> \circ' p &= <a, q \circ' p> \\ <a, p'> \circ' p &= <a, p' \circ p> \end{aligned}$$

b. $p_1 \cup p_2$ equals the union of the sets p_1 and p_2.

c. The mappings $\|: P \times P \to P$, $\lfloor\!\lfloor : P \times P \to P$, $\lfloor\!\lfloor' : Q \times P \to Q$ are given by

$$\begin{aligned} p_1 \| p_2 &= (p_1 \lfloor\!\lfloor p_2) \cup (p_2 \lfloor\!\lfloor p_1) \\ p_1 \lfloor\!\lfloor p_2 &= \{q \lfloor\!\lfloor' p_2 \mid q \in p_1\} \\ a \lfloor\!\lfloor' p &= <a, p> \\ <a, q> \lfloor\!\lfloor' p &= <a, q \lfloor\!\lfloor' p> \\ <a, p'> \lfloor\!\lfloor' p &= <a, p' \| p> \end{aligned}$$

Remark Note the essential difference between the last and the one but last clauses in parts a and c. E.g., in the last clause of c we change from leftmerge to (general) merge, in the one but last we repeat the leftmerge. (Cf. the similar difference in the operational setting explained earlier.)

Example
$\{<a, <b, \{c\}>>\} \| \{d\} = \{<a, <b, \{<c, \{d\}>\}>>, <a, <b, \{<d, \{c\}>\}>>, <d, \{<a, <b, \{c\}>>\}>\}$.

We have the following basic properties of $\circ$, $\|$ (the proofs of which rely on the rigorous definitions in the Appendix):

Lemma 4.2 $\circ$ is ndi in its first argument and contracting in its second argument; $\|$ and $\lfloor\!\lfloor$ are ndi in both arguments.
Proof See Appendix. □

In the equivalence proof ($\mathcal{O} = \mathcal{D}$) to be given below we shall use

Lemma 4.3

a. $\{<a, q> \mid q \in p_1\} \circ p_2 = \{<a, q> \mid q \in p_1 \circ p_2\}$.

b. $\{<a, q> \mid q \in p_1\} \lfloor\!\lfloor p_2 = \{<a, q> \mid q \in p_1 \lfloor\!\lfloor p_2\}$.

Proof See Appendix. □

Next, we introduce the operator *at* which will be used in the definition of the denotational meaning of $[s]$. We need a domain R which is essentially simpler than P in that it lacks the branching structure of P:

Definition 4.4 The domain R is the first component of the solution of the system of equations

$$\begin{aligned} R &\simeq \mathcal{P}_{nc}(T) \\ T &\simeq A \cup (A \times T/2) \end{aligned}$$

Each $r \in R$ consists of a (compact) set of elements which are either of the form $a \in A$ or of the form $<a, t>$ in $A \times T$. Note that R is nothing but the familiar linear-time or trace model: each r in R is a set of elements which are (equivalent to) words of one (case $a \in A$) or more (case $<a, t> \in A \times T$) symbols from A —including, by completeness, the case of infinite words over A. Clearly, R can be isometrically embedded into P. Where necessary, we shall use this without explicitly writing the mappings involved.

We now define $at: P \to R$ and $at': Q \to R$. $at(p)$ collapses the branching structure of p, and yields an element from R.

Definition 4.5

$$\begin{aligned} at(p) &= \bigcup\{at'(q) \mid q \in p\} \\ at'(a) &= \{a\} \\ at'(<a,q>) &= \{<a,t> \mid t \in at'(q)\} \\ at'(<a,p>) &= \{<a,t> \mid t \in at(p)\} \end{aligned}$$

A rigorous version of this definition is again provided in the Appendix.

Example $at(\{<a,\{b,c\}>\}) = at(\{<a,\{b\}>,<a,\{c\}>\}) = \{<a,b>,<a,c>\}$.

We have

Lemma 4.6

a. at is ndi.

b. $at(\{<a,q> \mid q \in p\}) = \{<a,t> \mid t \in at(p)\}$.

Proof Omitted (it requires the rigorous definitions of the Appendix). □

We are now sufficiently prepared for the definition of $\mathcal{D}_d: \mathcal{L}_{at} \to P$.

Definition 4.7

a. Let $F \in \mathcal{L}_{at} \to P$, and let $\Psi_d: (\mathcal{L}_{at} \to P) \to (\mathcal{L}_{at} \to P)$ be given by

$$\begin{aligned} \Psi_d(F)(a) &= \{a\} \\ \Psi_d(F)(x) &= \Psi_d(F)(d(x)) \\ \Psi_d(F)(s_1 ; s_2) &= \Psi_d(F)(s_1) \circ F(s_2) \\ \Psi_d(F)(s_1 + s_2) &= \Psi_d(F)(s_1) \cup \Psi_d(F)(s_2) \\ \Psi_d(F)(s_1 \parallel s_2) &= \Psi_d(F)(s_1) \parallel \Psi_d(F)(s_2) \\ \Psi_d(F)([s]) &= at(\Psi_d(F)(s)) \end{aligned}$$

b. $\mathcal{D}_d = fix(\Psi_d)$.

This definition is justified in

Lemma 4.8

a. $\Psi_d(F)$ is well-defined for each s.

b. Ψ_d is contracting in F.

Proof (sketch) Part a follows by induction on $c(s)$. Part b follows from the definition of $\Psi_d(F)$: For each s we show that $d(\Psi_d(F_1)(s), \Psi_d(F_2)(s)) \leq \frac{1}{2} d(F_1, F_2)$, using induction on $c(s)$ and Lemmas 4.2, 4.6a. The fact that $\circ$ is contracting in its second argument is the crucial step here (and explains the asymmetry in the definition of $\Psi_d(F)(s_1 ; s_2)$). Further details are provided in the Appendix. □

Remark For the purpose of comparing $\mathcal{O}$ and $\mathcal{D}$ it is necessary to extend $\mathcal{D}_d$ to $\mathcal{L}_{at}^+ \to P$, by replacing $\mathcal{L}_{at}$ by $\mathcal{L}_{at}^+$ in Definition 4.7 and adding the clause $\Psi_d(F)(s_1 \between s_2) = \Psi_d(F)(s_1) \between \Psi_d(F)(s_2)$.

We have reached the first main theorem of our paper, stating that, for each (d, s), $\mathcal{O}(d,s) = \mathcal{D}(d,s)$. We shall in fact show that $\mathcal{O}_d = \mathcal{D}_d$ (on $\mathcal{L}_{at}$), by establishing that $\Phi_d(\mathcal{D}_d) = \mathcal{D}_d$ (with Φ_d as in Definition 3.6). By the definition of $\mathcal{O}_d$ and Banach's theorem, the desired result follows. (This is another application of the proof method first proposed in [KR90].)

We first prove

Lemma 4.9

$$\begin{aligned}
\Phi_d(\mathcal{D}_d)(a) &= \{a\} \\
\Phi_d(\mathcal{D}_d)(x) &= \Phi_d(D_d)(d(x)) \\
\Phi_d(\mathcal{D}_d)(s_1 \,;\, s_2) &= \Phi_d(\mathcal{D}_d)(s_1) \circ \mathcal{D}_d(s_2) \\
\Phi_d(\mathcal{D}_d)(s_1 + s_2) &= \Phi_d(\mathcal{D}_d)(s_1) \cup \Phi_d(\mathcal{D}_d)(s_2) \\
\Phi_d(\mathcal{D}_d)(s_1 \parallel s_2) &= \Phi_d(\mathcal{D}_d)(s_1) \between \mathcal{D}(s_2) \cup \Phi_d(\mathcal{D}_d)(s_2) \between \mathcal{D}_d(s_1) \\
\Phi_d(\mathcal{D}_d)(s_1 \between s_2) &= \Phi_d(\mathcal{D}_d)(s_1) \between \mathcal{D}(s_2) \\
\Phi_d(\mathcal{D}_d)([s]) &= at(\Phi_d(\mathcal{D}_d)(s))
\end{aligned}$$

Proof We prove two typical subcases (we shall suppress indices d in the notation):
Case $s \equiv s_1 \,;\, s_2$:

$\Phi(\mathcal{D})(s_1 \,;\, s_2) =$
$\{a \mid s_1 \,;\, s_2 \xrightarrow{a}_0 \mathrm{E}\} \cup \{<a,q> \mid s_1 \,;\, s_2 \xrightarrow{a}_1 \tilde{s},\ q \in \mathcal{D}(\tilde{s})\} \cup \{<a, \mathcal{D}(\tilde{s})> \mid s_1 \,;\, s_2 \xrightarrow{a}_2 \tilde{s}\} =$
$\emptyset \cup \{<a,q> \mid s_1 \xrightarrow{a}_1 s',\ q \in \mathcal{D}(s' \,;\, s_2)\} \cup \{<a, \mathcal{D}(s_2)> \mid s_1 \xrightarrow{a}_0 \mathrm{E}\} \cup \{<a, \mathcal{D}(s' \,;\, s_2)> \mid s_1 \xrightarrow{a}_2 s'\} =$
$\{<a,q> \mid s_1 \xrightarrow{a}_1 s',\ q \in \mathcal{D}(s') \circ \mathcal{D}(s_2)\} \cup (\{a \mid s_1 \xrightarrow{a}_0 \mathrm{E}\} \circ \mathcal{D}(s_2)) \cup \{<a, \mathcal{D}(s') \circ \mathcal{D}(s_2)> \mid s_1 \xrightarrow{a}_2 s'\}$
$=$ (Lemma 4.3a)
$(\{a \mid s_1 \xrightarrow{a}_0 \mathrm{E}\} \cup \{<a,q> \mid s_1 \xrightarrow{a}_1 s',\ q \in \mathcal{D}(s')\} \cup \{<a, \mathcal{D}(s')> \mid s_1 \xrightarrow{a}_2 s'\}) \circ \mathcal{D}(s_2) =$
$\Phi(\mathcal{D})(s_1) \circ \mathcal{D}(s_2)$

Case $s \equiv [s_1]$:

$\Phi(\mathcal{D})([s_1]) =$
$\{a \mid [s_1] \xrightarrow{a}_0 \mathrm{E}\} \cup \{<a,q> \mid [s_1] \xrightarrow{a}_1 \tilde{s},\ q \in \mathcal{D}(\tilde{s})\} \cup \{<a, \mathcal{D}(\tilde{s})> \mid [s_1] \xrightarrow{a}_2 \tilde{s}\} =$
$\{a \mid s_1 \xrightarrow{a}_0 \mathrm{E}\} \cup \{<a,q> \mid s_1 \xrightarrow{a}_1 s',\ q \in \mathcal{D}([s'])\} \cup \{<a,q> \mid s_1 \xrightarrow{a}_2 s'',\ q \in \mathcal{D}([s''])\} \cup \emptyset =$
(by the definition of at and Lemma 4.6b)
$at(\{a \mid s_1 \xrightarrow{a}_0 \mathrm{E}\} \cup \{<a,q> \mid s_1 \xrightarrow{a}_1 s', q \in \mathcal{D}(s')\} \cup \{<a, \mathcal{D}(s'')> \mid s_1 \xrightarrow{a}_2 s''\}) =$
$at(\Phi(\mathcal{D})(s_1))$. □

We can now prove

Lemma 4.10 For each $s \in \mathcal{L}_{at}^+$, $\Phi_d(\mathcal{D}_d)(s) = \mathcal{D}_d(s)$.
Proof Induction on $c(s)$, using Lemma 4.9. E.g., $\Phi_d(\mathcal{D}_d)(s_1 \,;\, s_2) = \Phi_d(\mathcal{D}_d)(s_1) \circ \mathcal{D}(s_2) =$ (ind. hyp.) $\mathcal{D}_d(s_1) \circ \mathcal{D}_d(s_2) = \mathcal{D}_d(s_1 \,;\, s_2)$. □

Finally we have

Theorem 4.11 For each $s \in \mathcal{L}_{at}$, $\mathcal{O}_d(s) = \mathcal{D}_d(s)$.
Proof By the definition of $\mathcal{O}_d$, Lemma 4.10 and Banach's theorem. □

5 Action refinement

We extend $\mathcal{L}_{at}$ with the notion of *action refinement*, and call the extended language $\mathcal{L}_{atr}$. We discuss how to adapt $\mathcal{O}$ and $\mathcal{D}$ for $\mathcal{L}_{atr}$, and establish that $\mathcal{O} = \mathcal{D}$ remains valid.
The notion of action refinement, syntactically expressed by the construct $s<a \rightsquigarrow s'>$, refers to an operation in which all occurrences of a in s are 'refined' to s'. It is well-known (e.g., [Pra86, CMP87]) that this leads to problems in the semantic models which use the interleaving approach to concurrency: Consider the two statements $a \parallel b$ and $(a\ ;\ b) + (b\ ;\ a)$, which have the same meaning in any interleaving model. However, taking (for simplicity) a model where meanings are just sets of words, we expect to obtain as meaning for $(a \parallel b)<a \rightsquigarrow a_1;a_2>$ the set $\{a_1a_2b, a_1ba_2, ba_1a_2\}$, and for $((a\ ;\ b) + (b\ ;\ a))<a \rightsquigarrow a_1;a_2>$ the set $\{a_1a_2b, ba_1a_2\}$. This is clearly undesirable (it violates the compositionality principle); moreover, it is not difficult to verify that the same phenomenon would persist in our (P,Q)-domain (from Definition 3.5). As simple solution we propose to refine actions only by atomized statements: We take $s<a \rightsquigarrow s'>$ with the convention that s' is in fact atomized (i.e., we should write $s<a \rightsquigarrow [s']>$, but we drop the [,] not to burden the notation too much).

Below, we shall firstly add a few rules to $\mathcal{T}_{at}$ which conveniently handle refinement exploiting the already available machinery (with the $\rightarrow_i$ transitions) to deal with atomization. (Very similar rules were first presented in [Gor91]. See also [DG91].) Secondly, we propose a new denotational operator —which we might call semantic substitution—, written as $p<a \rightsquigarrow r>$: this operator replaces all a's in the (possibly infinite) process p by r (recall that meanings of any $[s]$ always reside in R). Moreover, we employ an auxiliary construct, viz. the *commit* operator (':'). This is like sequential composition, but in executing $s_1 : s_2$, no interleaving at the place of ':' is allowed.

We proceed with the precise definitions.

Definition 5.1

a. (syntax for $\mathcal{L}_{atr}$) We replace Definition 3.1 clause a by the definition

$$s ::= \ldots \mid s<a \rightsquigarrow s'>$$

(with ... as in Definition 3.1a).

b. ($\mathcal{L}^g_{atr}$) We replace Definition 3.1, clause b by the definition

$$g ::= \ldots \mid g<a \rightsquigarrow g'>$$

(with ... as in Definition 3.1b).

c. A program is a pair (d, s) with $d \in PVar \rightarrow \mathcal{L}^g_{atr}$ and $s \in \mathcal{L}_{atr}$.

We extend $\mathcal{L}_{atr}$ to $\mathcal{L}^+_{atr}$ which additionally includes statements of the format $s_1 : s_2$ (and also $s_1 \mathbin{\|\!\!\lfloor} s_2$) and put $\bar{\mathcal{L}}^+_{atr} = \mathcal{L}^+_{atr} \cup \{\mathrm{E}\}$. To the definition of the complexity measure c (Definition 3.3) we add the clause

$$c(s_1<a \rightsquigarrow s_2>) = c(s_1 : s_2) = c(s_1) + c(s_2) + 1$$

We now introduce the extended transition system $\mathcal{T}_{atr}$ which is obtained from $\mathcal{T}_{at}$ in the following way.

Definition 5.2 $\mathcal{T}_{atr}$ consists of the rules of $\mathcal{T}_{at}$ (where all d, s now refer to $\mathcal{L}^{+}_{atr}$) to which we add

(rules for refinement)

- $$\frac{s_1 \xrightarrow{b}_{i,d} \bar{s}}{s_1{<}a \rightsquigarrow s{>} \xrightarrow{b}_{i,d} \bar{s}{<}a \rightsquigarrow s{>}} \quad (a \neq b)$$ (refinement, 1)

- $$\frac{s_1 \xrightarrow{a}_{i,d} \bar{s} \quad , \quad [s] \xrightarrow{b}_{0,d} \mathrm{E}}{s_1{<}a \rightsquigarrow s{>} \xrightarrow{b}_{i,d} \bar{s}{<}a \rightsquigarrow s{>}}$$ (refinement, 2)

- $$\frac{s_1 \xrightarrow{a}_{0,d} \mathrm{E} \quad , \quad [s] \xrightarrow{b}_{1,d} s'}{s_1{<}a \rightsquigarrow s{>} \xrightarrow{b}_{1,d} s'}$$ (refinement, 3.1)

- $$\frac{s_1 \xrightarrow{a}_{1,d} \bar{s} \quad , \quad [s] \xrightarrow{b}_{1,d} s'}{s_1{<}a \rightsquigarrow s{>} \xrightarrow{b}_{1,d} s' : \bar{s}{<}a \rightsquigarrow s{>}}$$ (refinement, 3.2)

- $$\frac{s_1 \xrightarrow{a}_{2,d} \bar{s} \quad , \quad [s] \xrightarrow{b}_{1,d} s'}{s_1{<}a \rightsquigarrow s{>} \xrightarrow{b}_{1,d} s' \,;\, \bar{s}{<}a \rightsquigarrow s{>}}$$ (refinement, 3.3)

(rules for commit)

- $$\frac{s \xrightarrow{a}_{0,d} \mathrm{E}}{s : s_1 \xrightarrow{a}_{1,d} s_1} \qquad \frac{s \xrightarrow{a}_{1,d} s'}{s : s_1 \xrightarrow{a}_{1,d} s' : s_1} \qquad \frac{s \xrightarrow{a}_{2,d} s'}{s : s_1 \xrightarrow{a}_{2,d} s' : s_1}$$ (commit)

In case $\bar{s} \equiv \mathrm{E}$, it is understood that $\mathrm{E}{<}a \rightsquigarrow s{>} \equiv \mathrm{E}$.

Remark Note that, by the definition of $\mathcal{T}_{at}$, we firstly have that s' (in (refinement 3.1, 3.2, 3.3)) is itself of the form $[s'']$ for some suitable s''. The introduction of the ':' takes place in (refinement 3.2) induced by the $s_1 \xrightarrow{a}_1 \bar{s}$ premise. Also note that there is no point in distinguishing further cases with a premise $[s] \xrightarrow{b}_2 \ldots$, since such a transition is not derivable in $\mathcal{T}_{at}$ (or $\mathcal{T}_{atr}$). The rules for ':' are a natural counterpart of the rules for ';' as already present in $\mathcal{T}_{at}$. Note, however, that here we obtain a $\rightarrow_1$ step in the conclusion from a $\rightarrow_0$ step in the premise.

An explanation in words of the rules for $s_1{<}a \rightsquigarrow s{>}$ is as follows: If $s_1 \xrightarrow{b}_i \bar{s}$ for some $b \neq a$, the possible effects of a-refinement are simply transmitted to $\bar{s}$. If $s_1 \xrightarrow{a}_i \bar{s}$, then we see which steps (either $\rightarrow_0$ or $\rightarrow_1$) can be taken by the atomic $[s]$. In the case that $[s] \xrightarrow{b}_1 s'$, in the conclusion of the rules (refinement 3.1, 3.2, 3.3) we deliver a b-step which continues with execution of the statement s', $s' : \bar{s}{<}a \rightsquigarrow s{>}$ and $s' \,;\, \bar{s}{<}a \rightsquigarrow s{>}$, respectively. Note that the refinement affects all a's occurring in s, not just the first a-step taken by s. The explanation of the simpler (refinement 2) rule should be clear.

Remark An alternative treatment of refinement might be couched in terms of manipulating with procedures: Consider a program $(d, s_1{<}a{\rightsquigarrow}s{>})$. In order to obtain the desired effect, introduce some *fresh* x_a, and next

- replace all occurrences of a in s_1 by x_a, denoting the result by $s_1[^{x_a}/_a]$
- extend d to d' which is like d but for its value in x_a: $d'(x_a) = [s]$
- replace the program by $(d', s_1[^{x_a}/_a])$

Problematic with this 'solution' is

- the solution is obtained by changing the program rather than by the introduction of suitable transition rules
- the statement $[s]$ is not necessarily guarded
- it involves manipulating the declarations; much of the general semantic framework relies on the fact that d remains constant for a given program
- the refinement $x{<}a{\rightsquigarrow}s{>}$, where a occurs in $d(x)$, has no effect.

Just as in Definition 3.6, we define the higher-order operator Φ_d, now based on $\mathcal{T}_{atr}$, and again put $\mathcal{O}_d = fix(\Phi_d)$.

We continue with the denotational definitions. First we give the definitions of semantic substitution and of the semantic ':'.

Definition 5.3

a. The mappings $\cdot : \cdot : P \times P \to P$ and $\cdot :' \cdot : Q \times P \to P$ are given by

$$p_1 : p_2 = \bigcup\{ q_1 :' p_2 \mid q_1 \in p_1 \}$$

$$a :' p = \{ {<}a, q{>} \mid q \in p \}$$

$${<}a, q'{>} :' p = \{ {<}a, \bar{q}{>} \mid \bar{q} \in q' :' p \}$$

$${<}a, p'{>} :' p = \{{<}a, p' : p{>}\}$$

b. Let $a \in A$. The mappings $\cdot{<}a{\rightsquigarrow}\cdot{>} : P \times R \to P$ and $\cdot{<}a{\rightsquigarrow}'\cdot{>} : Q \times R \to P$ are given by

$$p{<}a{\rightsquigarrow}r{>} = \bigcup\{q{<}a{\rightsquigarrow}'r{>} \mid q \in p\}$$

$$b{<}a{\rightsquigarrow}'r{>} = \begin{cases} r & \text{if } a = b \\ \{b\} & \text{if } a \neq b \end{cases}$$

$${<}b, q{>}{<}a{\rightsquigarrow}'r{>} = \begin{cases} r : q{<}a{\rightsquigarrow}'r{>} & \text{if } a = b \\ \{b\} : q{<}a{\rightsquigarrow}'r{>} & \text{if } a \neq b \end{cases}$$

$${<}b, p{>}{<}a{\rightsquigarrow}'r{>} = \begin{cases} r \circ p{<}a{\rightsquigarrow}r{>} & \text{if } a = b \\ \{b\} \circ p{<}a{\rightsquigarrow}r{>} & \text{if } a \neq b \end{cases}$$

Again the above definitions may be made rigorous by the use of (simultaneous) higher-order mappings, cf. the Appendix for more details. In the construction of $\mathcal{D}$ in definition 5.6 below we need

Lemma 5.4 The functions $\cdot<a\rightsquigarrow\cdot>$ $(a \in A)$ and $\cdot : \cdot$ are ndi in both its arguments.

We have the following properties of the commit operator that will be needed in the proof of theorem 5.7.

Lemma 5.5

a. $\{<a,q> \mid q \in p_1\} : p_2 = \{<a,q> \mid q \in p_1 : p_2\}$

b. $t :' (\bigcup_{i\in I} p_i) = \bigcup_{i\in I}(t :' p_i)$ for any index set I such that $\bigcup_{i\in I} p_i \in P$

Proof See the Appendix. □

Remark Lemma 5.5b does not hold when we replace $t \in T$ by an arbitrary $q \in Q$.

Next, we define $\mathcal{D}_d$ for $\mathcal{L}_{atr}$:

Definition 5.6

a. We define the mapping $\Psi_d : (\mathcal{L}_{atr} \to P) \to (\mathcal{L}_{atr} \to P)$ by extending the clauses in Definition 4.7 with the new clause

$$\Psi_d(F)(s<a\rightsquigarrow s'>) = \Psi_d(F)(s)<a\rightsquigarrow\Psi_d(F)([s'])>$$

where $F \in \mathcal{L}_{atr} \to P$.

b. We put $\mathcal{D}_d = fix(\Psi_d)$ as before (but now referring to the extended Ψ_d).

It is not difficult to show that $\Psi_d(F)$ remains both well-defined and contracting in F. The proof uses the (extended) definition of c and Lemma 5.4. Similarly to what we did for Definition 4.7 we may extend $\mathcal{D}_d$ to $\mathcal{L}^+_{atr} \to P$ (with clauses for $s_1 \between s_2$ and $s_1 : s_2$).

Finally we state the main theorem of this Section, establishing that the equivalence of $\mathcal{O}$ and $\mathcal{D}$ is preserved by the extension of $\mathcal{L}_{at}$ with action refinement.

Theorem 5.7 $\mathcal{O}_d = \mathcal{D}_d$ on $\mathcal{L}_{atr}$.

Proof We follow the same argument as in Section 4, and now exhibit the proof that $\Phi_d(\mathcal{D}_d)(s_1<a\rightsquigarrow s>) = \mathcal{D}_d(s_1<a\rightsquigarrow s>)$. The proof of $\Phi_d(\mathcal{D}_d)(s_1 : s_2) = \mathcal{D}_d(s_1 : s_2)$ is similar to the corresponding proof for $s_1 ; s_2$ in Lemma 4.9 but now using Lemma 5.5a. We'll drop the subscripts d as before; we abbreviate, where convenient $\mathcal{D}([s])$ to r.
Firstly we have, by the (adapted) definition of Φ, that

$$\Phi(\mathcal{D})(s_1<a\rightsquigarrow s>) = (1) \cup (2) \cup \cdots \cup (9)$$

where

$$
\begin{aligned}
(1) &= \{b \mid s_1 \xrightarrow{b}_0 \mathrm{E},\ a \neq b\} \\
(2) &= \{b \mid s_1 \xrightarrow{a}_0 \mathrm{E},\ [s] \xrightarrow{b}_0 \mathrm{E}\} \\
(3) &= \{<b,q> \mid s_1 \xrightarrow{b}_1 s',\ a \neq b,\ q \in \mathcal{D}(s'<a \rightsquigarrow s>)\} \\
(4) &= \{<b,q> \mid s_1 \xrightarrow{a}_1 \bar{s},\ [s] \xrightarrow{b}_0 \mathrm{E},\ q \in \mathcal{D}(\bar{s}<a \rightsquigarrow s>)\} \\
(5) &= \{<b,q> \mid s_1 \xrightarrow{a}_0 \mathrm{E},\ [s] \xrightarrow{b}_1 s',\ q \in \mathcal{D}(s')\} \\
(6) &= \{<b,q> \mid s_1 \xrightarrow{a}_1 \bar{s},\ [s] \xrightarrow{b}_1 s',\ q \in \mathcal{D}(s' : \bar{s}<a \rightsquigarrow s>)\} \\
(7) &= \{<b,q> \mid s_1 \xrightarrow{a}_2 \bar{s},\ [s] \xrightarrow{b}_1 s',\ q \in \mathcal{D}(s' ; \bar{s}<a \rightsquigarrow s>)\} \\
(8) &= \{<b,\mathcal{D}(\bar{s}<a \rightsquigarrow s>)> \mid s_1 \xrightarrow{b}_2 \bar{s},\ a \neq b\} \\
(9) &= \{<b,\mathcal{D}(\bar{s}<a \rightsquigarrow s>)> \mid s_1 \xrightarrow{a}_2 \bar{s},\ [s] \xrightarrow{b}_0 \mathrm{E}\}
\end{aligned}
$$

Moreover, we have that

$$
\begin{aligned}
&\mathcal{D}(s_1<a \rightsquigarrow s>) \\
&\quad = \mathcal{D}(s_1)<a \rightsquigarrow r> \\
&\quad = \text{(induction hypothesis for } s_1) \\
&\qquad (\{\, b \mid s_1 \xrightarrow{b}_0 \mathrm{E}\,\} \cup \{\, <b,q> \mid s_1 \xrightarrow{b}_1 \bar{s},\ q \in \mathcal{D}(\bar{s})\,\} \cup \{\, <b,\mathcal{D}(\bar{s})> \mid s_1 \xrightarrow{b}_2 \bar{s}\,\})<a \rightsquigarrow r> \\
&\quad = (1') \cup (2') \cup \cdots \cup (6')
\end{aligned}
$$

where

$$
\begin{aligned}
(1') &= \bigcup\{r \mid s_1 \xrightarrow{a}_0 \mathrm{E}\} \\
(2') &= \{b \mid s_1 \xrightarrow{b}_0 \mathrm{E},\ a \neq b\} \\
(3') &= \bigcup\{r : q<a \rightsquigarrow' r> \mid s_1 \xrightarrow{a}_1 \bar{s},\ q \in \mathcal{D}(\bar{s})\} \\
(4') &= \{<b,q'> \mid s_1 \xrightarrow{b}_1 \bar{s},\ q \in \mathcal{D}(\bar{s}),\ q' \in q<a \rightsquigarrow' r>,\ a \neq b\} \\
(5') &= \bigcup\{r \circ \mathcal{D}(\bar{s})<a \rightsquigarrow r> \mid s_1 \xrightarrow{a}_2 \bar{s}\} \\
(6') &= \{<b,\mathcal{D}(\bar{s})<a \rightsquigarrow r>> \mid s_1 \xrightarrow{b}_2 \bar{s},\ a \neq b\}
\end{aligned}
$$

We have $(2') = (1)$, $(4') = (3)$, $(6') = (8)$ and by the induction hypothesis

$$
\begin{aligned}
(1') &= \{b \mid [s] \xrightarrow{b}_0 \mathrm{E},\ s_1 \xrightarrow{a}_0 \mathrm{E}\} \cup \{<b,q'> \mid [s] \xrightarrow{b}_1 s',\ q' \in \mathcal{D}(s'),\ s_1 \xrightarrow{a}_0 \mathrm{E}\} \\
&= (2) \cup (5)
\end{aligned}
$$

Next we show $(3') = (4) \cup (6)$. First suppose $[s] \xrightarrow{b}_1 s'$. By inspection of the transition system follows $s' = [s'']$ for some suitable s''. Therefore it holds that $\mathcal{D}(s') \in R$ and

$$
\begin{aligned}
&\mathcal{D}(s' : \bar{s}<a \rightsquigarrow s>) \\
&\quad = \text{(definition of } \mathcal{D}) \\
&\qquad \mathcal{D}(s') : \mathcal{D}(\bar{s})<a \rightsquigarrow r> \\
&\quad = \text{(definition :, } \mathcal{D}(s') \in R) \\
&\qquad \bigcup\{\, t' :' \mathcal{D}(\bar{s})<a \rightsquigarrow r> \mid t' \in \mathcal{D}(s')\} \\
&\quad = \text{(definition } <a \rightsquigarrow r>) \\
&\qquad \bigcup\{\, t' :' (\bigcup\{\bar{q}<a \rightsquigarrow' r> \mid \bar{q} \in \mathcal{D}(\bar{s})\}) \mid t' \in \mathcal{D}(s')\,\} \\
&\quad = \text{(Lemma 5.5b)} \\
&\qquad \bigcup\{\, q' :' \bar{q}<a \rightsquigarrow' r> \mid \bar{q} \in \mathcal{D}(\bar{s}),\ q' \in \mathcal{D}(s')\,\}
\end{aligned}
$$

Now we reason as follows

$(3')$ = (induction hypothesis, definition Φ)
$\bigcup\{(\{b \mid [s] \xrightarrow{b}_0 \mathrm{E}\} \cup \{<b,q'> \mid [s] \xrightarrow{b}_1 s',\ q' \in \mathcal{D}(s')\}) : \bar{q}<a \rightsquigarrow' r> \mid s_1 \xrightarrow{a}_1 \bar{s},\ \bar{q} \in \mathcal{D}(\bar{s})\}$
= (definition :)
$\{<b,\tilde{q}> \mid \tilde{q} \in \bar{q}<a \rightsquigarrow' r>,\ [s] \xrightarrow{b}_0 \mathrm{E},\ s_1 \xrightarrow{a}_1 \bar{s},\ \bar{q} \in \mathcal{D}(\bar{s})\} \cup$
$\{<b,\tilde{q}> \mid \tilde{q} \in q' : \bar{q}<a \rightsquigarrow' r>,\ q' \in \mathcal{D}(s'),\ [s] \xrightarrow{b}_1 s',\ s_1 \xrightarrow{a}_1 \bar{s},\ \bar{q} \in \mathcal{D}(\bar{s})\}$
= (by the above)
$(4) \cup (6)$

Similarly, we have by the definitions of $\mathcal{D}$ and $\circ$

$$\mathcal{D}(s' ; \bar{s}<a \rightsquigarrow s>) = \{q' \circ \mathcal{D}(\bar{s}<a \rightsquigarrow s>) \mid q' \in \mathcal{D}(s')\}$$

and therefore

$(5')$ = (induction hypothesis)
$\bigcup\{(\{b \mid [s] \xrightarrow{b}_0 \mathrm{E}\} \cup \{<b,q'> \mid [s] \xrightarrow{b}_1 s',\ q' \in \mathcal{D}(s')\}) \circ \mathcal{D}(\bar{s}<a \rightsquigarrow r>) \mid s_1 \xrightarrow{a}_2 \bar{s}\}$
= (Lemma 4.3a)
$\{<b, \mathcal{D}(\bar{s}<a \rightsquigarrow r>) \mid [s] \xrightarrow{b}_0 s',\ s_1 \xrightarrow{a}_2 \bar{s}\} \cup$
$\{<b, q' \circ \mathcal{D}(s<a \rightsquigarrow r>) \mid [s] \xrightarrow{b}_1 s',\ q' \in \mathcal{D}(s'),\ s_1 \xrightarrow{a}_2 \bar{s}\}$
= $(9) \cup (7)$

Altogether, we have proven that $\Phi(\mathcal{D})(s_1<a \rightsquigarrow s>) = \mathcal{D}(s_1<a \rightsquigarrow s>)$ as desired. □

Concluding remarks

1. It is possible to design an alternative transition system for action refinement based on syntactic substitution (we owe this idea to Jan Rutten). Let us write $s_1 \rightarrow s_2$ as short hand for the rule

$$\frac{s_2 \xrightarrow{a}_{i,d} \bar{s}}{s_1 \xrightarrow{a}_{i,d} \bar{s}}$$

for some a and $\bar{s}$, and any $i = 0, 1, 2$.

We define the new $\mathcal{T}'_{atr}$ to consist of $\mathcal{T}_{at}$, together with the rules $a<a \rightsquigarrow s'> \rightarrow [s']$, $b<a \rightsquigarrow s'> \rightarrow b$ $(a \neq b)$, $(s_1 \underline{op}\, s_2)<a \rightsquigarrow s'> \rightarrow s_1<a \rightsquigarrow s'> \underline{op}\, s_2<a \rightsquigarrow s'>$, $\underline{op} \in \{;, +, \|\}$, $x<a \rightsquigarrow s'> \rightarrow d(x)<a \rightsquigarrow s'>$, and $[s]<a \rightsquigarrow s'> \rightarrow [s<a \rightsquigarrow s'>]$. Thus, we deal with $s<a \rightsquigarrow s'>$ by a syntactic substitution directed by the structure of s. Note that the above rules are not yet complete: The case $s \equiv s_0<b \rightsquigarrow s'>$ is lacking. To remedy this, one would require the introduction of simultaneous refinement, say $s<a_i \rightsquigarrow s_i>_{i=1}^n$, together with a rule for $s<a_i \rightsquigarrow s_i>_{i=1}^n <b_j \rightsquigarrow s'_j>_{j=1}^m$ (details omitted). After having taken care of this, we may introduce $\mathcal{O}'$ from $\mathcal{T}'_{atr}$ in the usual way, and the natural question as to how to prove $\mathcal{O}' = \mathcal{D}$ arises. A proof in the style as used in that of Theorem 5.7 is, in principle, possible. However, nontrivial *semantic* properties of refinement, such as compositionality results of the form $(p_1 \underline{op}\, p_2)<a \rightsquigarrow r> = p_1<a \rightsquigarrow r> \underline{op}\, p_2<a \rightsquigarrow r>$ $(\underline{op} \in \{\circ, \|, \cup\})$ are then required, and a full elaboration of this demands considerable work. Since, moreover, the present system $\mathcal{T}_{atr}$ may be claimed to provide a good deal more *operational* insight into the effect of the refinement operation, we have preferred it over the alternative $\mathcal{T}'_{atr}$.

2. In work in progress (continuing [Rut90b, Rut90c]), Rutten and Turi are studying a general method to obtain (not only $\mathcal{O}$ but also) $\mathcal{D}$ from a transition system $\mathcal{T}$, provided this system satisfies certain syntactic restrictions (of the kind first discussed in [GV89]), and then to prove $\mathcal{O} = \mathcal{D}$. Clearly, if a general result of this form would be available it would obviate the need for many detailed equivalence proofs, including the one just given for Theorem 5.7.

References

[Ace90] L. Aceto. *Action Refinement in Process Algebras.* PhD thesis, University of Sussex, 1990.

[ABKR89] P.H.M. America, J.W. de Bakker, J.N. Kok, and J.J.M.M. Rutten. Denotational semantics of a parallel object-oriented language. *Information and Computation*, 83:152–205, 1989.

[AR89] P. America and J.J.M.M. Rutten. Solving reflexive domain equations in a category of complete metric spaces. *Journal of Computer and System Sciences*, 39:343–375, 1989.

[B91] J.W. de Bakker. Comparative semantics for flow of control in logic programming without logic. *Information and Computation*, 94:123–179, 1991.

[BBKM84] J.W. de Bakker, J.A. Bergstra, J.W. Klop, and J.-J.Ch. Meyer. Linear time and branching time semantics for recursion with merge. *Theoretical Computer Science*, 34:135–156, 1984.

[BK90] J.W. de Bakker and J.N. Kok. Comparative semantics for Concurrent Prolog. *Theoretical Computer Science*, 75:15–44, 1990.

[BW90] J.W. de Bakker and J.H.A. Warmerdam. Metric pomset semantics for a concurrent language with recursion. In I. Guessarian, editor, *Semantics of Systems of Concurrent Processes*, pages 21–49. LNCS 469, Springer-Verlag, 1990.

[BW91] J.W. de Bakker and J.H.A. Warmerdam. Four domains for concurrency. *Theoretical Computer Science*, 90:127–149, 1991.

[BZ82] J.W. de Bakker and J.I. Zucker. Processes and the denotational semantics of concurrency. *Information and Control*, 54:70–120, 1982.

[BW90] J.C.M. Baeten and W.P. Weijland. *Process Algebra.* Cambridge University Press, 1990.

[BK84] J.A. Bergstra and J.W. Klop. Process algebra for synchronous communication. *Information and Control*, 60:109–137, 1984.

[BGV91] W. Brauer, R. Gold, and W. Vogler. A survey of behaviour and equivalence preserving refinements of Petri nets. In G. Rozenberg, editor, *Advances in Petri Nets*, pages 1–46. LNCS 483, Springer-Verlag, 1991.

[CMP87] L. Castellano, G. de Michelis, and L. Pomello. Concurrency vs. interleaving: An instructive example. *Bulletin of the EATCS*, 31:12–15, 1987.

[DG91] P. Degano and R. Gorrieri. Atomic refinement in process description languages. In A. Tarlecki, editor, *Mathematical Foundations of Computer Science*, pages 121–130. LNCS 520, Springer-Verlag, 1991.

[Gla90] R.J. van Glabbeek. *Comparative Concurrency Semantics and Refinement of Actions.* PhD thesis, Vrije Universiteit, Amsterdam, 1990.

[GR89] R.J. van Glabbeek and J.J.M.M. Rutten. The processes of De Bakker and Zucker represent bisimulation equivalence classes. In *J.W. de Bakker, 25 Jaar Semantiek, Liber Amicorum*, pages 243–246. CWI, Amsterdam, 1989.

[Gor91] R. Gorrieri. *Refinement, Atomicity and Transactions for Process Description Languages.* PhD thesis, University of Pisa, 1991.

[GV89] J.F. Groote and F.W. Vaandrager. Structured operational semantics and bisimulation as a congruence. In G. Ausiello, M. Dezani-Ciancaglini, and S. Ronchi Della Rocca, editors, *Proc. ICALP'89*, pages 423–438. LNCS 372, Springer-Verlag, 1989.

[Hah48] H. Hahn. *Reelle Funktionen.* Chelsea, 1948.

[HBR90] E. Horita, J.W. de Bakker, and J.J.M.M. Rutten. Fully abstract denotational semantics for nonuniform concurrent languages. Technical Report CS-R9027, CWI, Amsterdam, 1990. To appear in Information and Computation.

[KK90] P. Knijnenburg and J.N. Kok. Semantic models for parallel choice in combination with atomicity. Technical report, Dept. of Computer Science, State University of Utrecht, 1990.

[KK91] P. Knijnenburg and J.N. Kok. On the semantics of atomized statements: the parallel choice option. In L. Budach, editor, *Fundamentals of Computation Theory.* LNCS 529, Springer-Verlag, 1991.

[KR90] J.N. Kok and J.J.M.M. Rutten. Contractions in comparing concurrency semantics. *Theoretical Computer Science*, 76:180–222, 1990.

[Kur56] K. Kuratowski. Sur une méthode de métrisation complète des certains espaces d'ensembles compacts. *Fundamenta Mathematicae*, 42:114–138, 1956.

[Mic51] E. Michael. Topologies on spaces of subsets. *Transactions of the AMS*, 71:152–182, 1951.

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

[Pra86] V. Pratt. Modelling concurrency with partial orders. *International Journal of Parallel Programming*, 15:33–71, 1986.

[Rut89] J.J.M.M. Rutten. Correctness and full abstraction of metric semantics for concurrency. In J.W. de Bakker, W.P. de Roever, and G. Rozenberg, editors, *Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency*, pages 628–659. LNCS 354, Springer-Verlag, 1989.

[Rut90a] J.J.M.M. Rutten. Semantic correctness for a parallel object-oriented language. *SIAM Journal on Computing*, 19:341–383, 1990.

[Rut90b] J.J.M.M. Rutten. Deriving denotational models for bisimulation from structured operational semantics. In M. Broy, and C.B. Jones, editors, *Proc. of the IFIP Working Group 2.2/2.3 Working Conference*, Sea of Galilee, pages 155–177. North-Holland, 1990.

[Rut90c] J.J.M.M. Rutten. Non well-founded sets and programming language semantics. Technical Report CS-R9063, CWI, Amsterdam, 1990. To appear in Proc. Mathematical Foundations of Programming Semantics, Pittsburgh, March 1991.

Appendix

We discuss a sample of topics not dealt with fully rigorously above. We first consider the definition of the semantic operators $\circ$, $\|$ and *at* introduced in Section 3. We show that they can be defined in terms of higher-order mappings, and that they satisfy the desired properties.

Definition A.1 Let us use the notation $(\phi \in)\, \mathbb{P} = P \times P \to^1 P$, and $(\psi \in)\, \mathbb{Q} = Q \times P \to^1 Q$. We define the mappings $\Omega_\circ$, $\Omega_\|$: $\mathbb{P} \times \mathbb{Q} \to \mathbb{P}$ in terms of the auxiliary $\Omega_{\circ'}$: $\mathbb{P} \times \mathbb{Q} \to \mathbb{Q}$ by putting

a.
$$\begin{aligned}
\Omega_\circ(\phi,\psi)(p_1,p_2) &= \{\, \Omega_{\circ'}(\phi,\psi)(q,p_2) \mid q \in p_1 \,\} \\
\Omega_\|(\phi,\psi)(p_1,p_2) &= \Omega_\circ(\phi,\psi)(p_1,p_2) \cup \Omega_\circ(\phi,\psi)(p_2,p_1) \\
\Omega_{\circ'}(\phi,\psi)(a,p) &= <a,p> \\
\Omega_{\circ'}(\phi,\psi)(<a,q>,p) &= <a,\psi(q,p)> \\
\Omega_{\circ'}(\phi,\psi)(<a,p'>,p) &= <a,\phi(p',p)>
\end{aligned}$$

b. $<\circ,\circ'> = fix(<\Omega_\circ,\Omega_{\circ'}>)$, $<\|,\between'> = fix(<\Omega_\|,\Omega_{\circ'}>)$, $\between = \Omega_\circ(\|,\between')$

These definitions are justified by

Lemma A.2

a. $\Omega_{\circ'}(\phi,\psi) \in \mathbb{Q}$, $\Omega_\circ(\phi,\psi)$, $\Omega_\|(\phi,\psi) \in \mathbb{P}$.

b. $<\Omega_\circ,\Omega_{\circ'}>$ and $<\Omega_\|,\Omega_{\circ'}>$ are contracting in $<\phi,\psi>$.

Proof

a. We omit the easy proof that, for each ϕ, ψ the result $\Omega_{\circ'}(\phi,\psi)$ is in $\mathbb{Q}$. To show that $\Omega_\circ(\phi,\psi) \in \mathbb{P}$ we reason as follows: Take any p_1, p_2. Since $\Omega_\circ(\phi,\psi)(p_1,p_2) = \{\, \Omega_{\circ'}(\phi,\psi)(q,p_2) \mid q \in p_1 \,\}$, from Lemma 2.11 and the fact that $\Omega_{\circ'}(\phi,\psi)$ is ndi in q and p_2, we infer that $\Omega_\circ(\phi,\psi)$ is ndi in p_1, p_2. Additionally, by an appeal to Theorem 2.10, $\Omega_\circ(\phi,\psi)(p_1,p_2)$ is compact, since p_1 and $\Omega_{\circ'}(\phi,\psi)(q,p_2)$ (all $q \in p_2$) are compact and $\Omega_{\circ'}(\phi,\psi)$ is ndi, hence continuous. Well-definedness of $\Omega_\|$ now follows directly from its definition and Lemma 2.9c.

b. Immediate by the definitions of $<\Omega_\circ,\Omega_{\circ'}>$ and $<\Omega_\|,\Omega_{\circ'}>$. □

For the well-definedness of $\mathcal{D}$ (Lemma A.11) we need

Lemma A.3

a. $\circ$ is ndi in its first and $\frac{1}{2}$-contracting in its second argument.

b. $\|$ and $\between$ are ndi in both arguments.

Proof

a. Define $X \subseteq \mathbb{P} \times \mathbb{Q}$ by

$$X = \{ <\phi, \psi> \mid \phi, \psi \text{ ndi in first, } \tfrac{1}{2}\text{-contracting in second argument} \}$$

Note that X is a nonempty and closed subset of the complete metric space $\mathbb{P} \times \mathbb{Q}$. Moreover, it is straightforward, using Lemma 2.9, to check $<\Omega_\circ, \Omega_{\circ'}>(X) \subseteq X$. Hence by Lemma 2.11 it follows that $<\circ, \circ'> = fix<\Omega_\circ, \Omega_{\circ'}> \in X$.

b. For $\|$, by a similar argument as in part a for $<\Omega_\|, \Omega_{\circ'}>$. For $\between$ it subsequently follows from the definitions. □

The following simple properties of $\circ$, $\between$ were mentioned in Lemma 4.3.

Lemma A.4

a. $\{<a, q> \mid q \in p_1\} \circ p_2 = \{<a, q> \mid q \in p_1 \circ p_2\}$.

b. $\{<a, q> \mid q \in p_1\} \between p_2 = \{<a, q> \mid q \in p_1 \between p_2\}$.

Proof

a. $\{ <a, \bar{q}> \mid \bar{q} \in p_1 \} \circ p_2 = \{ <a, \bar{q}> \circ' p_2 \mid \bar{q} \in p_1 \} = \{ <a, \bar{q} \circ' p_2> \mid \bar{q} \in p_1 \} =$
$\{ <a, q> \mid q \in \{\bar{q} \circ' p_2 \mid \bar{q} \in p_1\} \} = \{ <a, q> \mid q \in p_1 \circ p_2 \}$.

b. Similar. □

We continue with a discussion of the at-operator.

Definition A.5 Let $(\phi \in) \mathbb{P}' \stackrel{df}{=} P \to^1 R$, $(\psi \in) \mathbb{Q}' \stackrel{df}{=} Q \to^1 R$.

a. We define the mappings $\Omega_{at}: \mathbb{P}' \times \mathbb{Q}' \to \mathbb{P}'$, and $\Omega_{at'}: \mathbb{P}' \times \mathbb{Q}' \to \mathbb{Q}'$, by putting

$$\begin{aligned}
\Omega_{at}(\phi, \psi)(p) &= \bigcup\{ \Omega_{at'}(\phi, \psi)(q) \mid q \in p \} \\
\Omega_{at'}(\phi, \psi)(a) &= \{a\} \\
\Omega_{at'}(\phi, \psi)(<a, q>) &= \{ <a, t> \mid t \in \psi(q) \} \\
\Omega_{at'}(\phi, \psi)(<a, p>) &= \{ <a, t> \mid t \in \phi(p) \}
\end{aligned}$$

b. $<at, at'> = fix(<\Omega_{at}, \Omega_{at'}>)$.

This definition is justified in

Lemma A.6

a. $\Omega_{at'}(\phi,\psi) \in \mathbb{Q}'$, $\Omega_{at}(\phi,\psi) \in \mathbb{P}'$.

b. $<\Omega_{at}, \Omega_{at'}>$ is contracting in $<\phi,\psi>$.

c. $at \in \mathbb{P}'$, $at' \in \mathbb{Q}'$.

Proof

a. Everything is direct from the definitions, apart from the compactness of the results. For $\Omega_{at'}$ this is easy: take any $\phi \in \mathbb{P}'$, $\psi \in \mathbb{Q}'$. Clearly, $\{a\}$, $\{<a,t> \mid t \in \psi(q)\} = \{a\} \times \psi(q)$, and $\{<a,t> \mid t \in \phi(p)\} = \{a\} \times \phi(p)$ are compact subsets of R, by the compactness of $\psi(q)$ and $\phi(p)$. In order to prove compactness of $\Omega_{at}(\phi,\psi)(p)$ we apply Theorem 2.10: each $\Omega_{at'}(\phi,\psi)(q)$ is a compact subset of R; moreover $\{\Omega_{at'}(\phi,\psi)(q) \mid q \in p\}$, the image of the set p under $\Omega_{at'}(\phi,\psi)$, is compact by continuity of $\Omega_{at'}(\phi,\psi)$ (Lemma A.2) and compactness of p. Hence $\Omega_{at}(\phi,\psi)(p)$ is compact.

b. Direct from the definitions.

c. Similar to Lemma A.3. □

The proof of the next lemma is similar to that of Lemma A.4.

Lemma A.7 $at(\{<a,q> \mid q \in p\}) = \{<a,t> \mid t \in at(p)\}$.
Proof Omitted. □

Next we turn to the definition of the commit and of the refinement.

Definition A.8

a. Let $(\phi \in)\tilde{\mathbb{P}} \stackrel{df}{=} P \times P \to^1 P$ and $(\psi \in)\tilde{\mathbb{Q}} \stackrel{df}{=} Q \times P \to^1 P$. The mappings $\Omega_{:}: \tilde{\mathbb{P}} \times \tilde{\mathbb{Q}} \to \tilde{\mathbb{P}}$ and $\Omega_{:'}: \tilde{\mathbb{P}} \times \tilde{\mathbb{Q}} \to \tilde{\mathbb{Q}}$ are given by

$$\begin{aligned}
\Omega_{:}(\phi,\psi)(p_1,p_2) &= \bigcup\{\Omega_{:'}(\phi,\psi)(q,p_2) \mid q \in p_1\} \\
\Omega_{:'}(\phi,\psi)(a,p) &= \{<a,q> \mid q \in p\} \\
\Omega_{:'}(\phi,\psi)(<a,q>,p) &= \{<a,\bar{q}> \mid \bar{q} \in \psi(q,p)\} \\
\Omega_{:'}(\phi,\psi)(<a,p'>,p) &= \{<a,\phi(p',p)>\}
\end{aligned}$$

Put $<:,:'> = fix(<\Omega_{:},\Omega_{:'}>)$.

b. Let $(\phi \in)\hat{\mathbb{P}} \stackrel{df}{=} P \times R \to^1 P$, $(\psi \in)\hat{\mathbb{Q}} \stackrel{df}{=} Q \times R \to^1 P$ and $a \in A$. The mappings $\Omega_{\rightsquigarrow}: \hat{\mathbb{P}} \times \hat{\mathbb{Q}} \to \hat{\mathbb{P}}$ and $\Omega_{\rightsquigarrow'}: \hat{\mathbb{P}} \times \hat{\mathbb{Q}} \to \hat{\mathbb{Q}}$ are given by

$$\Omega_{\rightsquigarrow}(\phi,\psi)(p,r) = \bigcup\{\Omega_{\rightsquigarrow'}(\phi,\psi)(q,r) \mid q \in p\}$$

$$\Omega_{\rightsquigarrow'}(\phi,\psi)(b,r) = \begin{cases} r & \text{if } a = b \\ \{b\} & \text{if } a \neq b \end{cases}$$

$$\Omega_{\rightsquigarrow'}(\phi,\psi)(<b,q>,r) = \begin{cases} r : \psi(q,r) & \text{if } a = b \\ \{b\} : \psi(q,r) & \text{if } a \neq b \end{cases}$$

$$\Omega_{\rightsquigarrow'}(\phi,\psi)(<b,p>,r) = \begin{cases} r \circ \phi(p,r) & \text{if } a = b \\ \{b\} \circ \phi(p,r) & \text{if } a \neq b \end{cases}$$

We put $<\cdot<a\rightsquigarrow\cdot>, \cdot<a\rightsquigarrow'\cdot>> = fix(<\Omega_{\rightsquigarrow}, \Omega_{\rightsquigarrow'}>)$.

Well-definedness of Definition A.8 (and Definition 5.3) follows from the next lemma. Moreover, part c of the lemma is needed in Lemma A.11.

Lemma A.9

a. $\Omega_{:'}(\phi,\psi) \in \tilde{\mathbb{Q}}$, $\Omega_{:}(\phi,\psi) \in \tilde{\mathbb{P}}$, $\Omega_{\rightsquigarrow'}(\phi,\psi) \in \hat{\mathbb{Q}}$, $\Omega_{\rightsquigarrow}(\phi,\psi) \in \hat{\mathbb{P}}$.

b. $<\Omega_{:}, \Omega_{:'}>$ and $<\Omega_{\rightsquigarrow}, \Omega_{\rightsquigarrow'}>$ are contracting in $<\phi,\psi>$.

c. $\cdot:\cdot$ and $\cdot<a\rightsquigarrow\cdot>$ are ndi in both arguments.

Proof

a. By now straightforward from the definitions, application of Theorem 2.10 and Lemma 2.11.

b. Directly from the definitions.

c. Similar to Lemma A.3. □

The properties stated in Lemma 5.5 are treated in

Lemma A.10

a. $\{<a,q> \mid q \in p_1\} : p_2 = \{<a,q> \mid q \in p_1 : p_2\}$

b. $t :' (\bigcup_{i\in I} p_i) = \bigcup_{i\in I} (t :' p_i)$ for any index set I such that $\bigcup_{i\in I} p_i \in P$

Proof

a. Straightforward from the definitions.

b. Similar to A.3 using the nonempty closed subset

$$X = \{\, <\phi,\psi> \mid \phi(r, \bigcup_i p_i) = \bigcup_i \phi(r,p_i)\,,\ \psi(t, \bigcup_i p_i) = \bigcup_i \psi(t,p_i)\,\}$$

of $\tilde{\mathbb{P}} \times \tilde{\mathbb{Q}}$. □

We conclude this Appendix with a discussion on the well-definedness of $\mathcal{D}_d$. It is sufficient to prove contractiveness of Ψ_d (given in Definition 4.7, 5.5):

Lemma A.11 $\Psi_d \in (\mathcal{L}_{atr} \to P) \to^{\frac{1}{2}} (\mathcal{L}_{atr} \to P)$

Proof We show that, for each $s \in \mathcal{L}_{atr}$,

$$d(\Psi_d(F_1)(s), \Psi_d(F_2)(s)) \leq \tfrac{1}{2} d(F_1, F_2)$$

We use induction on $c(s)$. We consider two typical subcases.
Case $s \equiv s_1 \,;\, s_2$:

$$\begin{array}{lll} & d(\Psi_d(F_1)(s_1\,;\,s_2), \Psi_d(F_2)(s_1\,;\,s_2)) & \\ = & d(\Psi_d(F_1)(s_1) \circ F_1(s_2), \Psi_d(F_2)(s_1) \circ F_2(s_2)) & \\ \leq & (d \text{ is an ultrametric}) & \\ & \max\{\ d(\Psi_d(F_1)(s_1) \circ F_1(s_2), \Psi_d(F_1)(s_1) \circ F_2(s_2)) & (=(*)), \\ & \qquad d(\Psi_d(F_1)(s_1) \circ F_2(s_2), \Psi_d(F_2)(s_1) \circ F_2(s_2)) & (=(**))\ \} \end{array}$$

We have

$$(*) \leq \tfrac{1}{2} d(F_1(s_2), F_2(s_2)) \leq \tfrac{1}{2} d(F_1, F_2)$$

since $\circ$ is contracting in its second argument and

$$(**) \leq \tfrac{1}{2} d(F_1, F_2)$$

since $\circ$ is ndi in its first argument by the induction hypothesis.
Case $s \equiv s_1 \langle a \rightsquigarrow s_2 \rangle$:

$$\begin{aligned}
& d(\Psi_d(F_1)(s_1\langle a \rightsquigarrow s_2\rangle), \Psi_d(F_2)(s_1\langle a \rightsquigarrow s_2\rangle)) \\
&\quad = \; d(\Psi_d(F_1)(s_1)\langle a \rightsquigarrow \Psi_d(F_1)([s_2])\rangle, \Psi_d(F_2)(s_1)\langle a \rightsquigarrow \Psi_d(F_2)([s_2])\rangle) \\
&\quad \leq \; (d \text{ an ultrametric}) \\
&\qquad \max\{\; d(\Psi_d(F_1)(s_1)\langle a \rightsquigarrow at(\Psi_d(F_1)(s_2))\rangle, \Psi_d(F_1)(s_1)\langle a \rightsquigarrow at(\Psi_d(F_2)(s_2))\rangle) \quad (=(*)), \\
&\qquad\qquad d(\Psi_d(F_1)(s_1)\langle a \rightsquigarrow at(\Psi_d(F_2)(s_2))\rangle, \Psi_d(F_2)(s_1)\langle a \rightsquigarrow at(\Psi_d(F_2)(s_2))\rangle) \quad (=(**))\;\}
\end{aligned}$$

We have

$$(*) \leq \tfrac{1}{2} d(F_1, F_2)$$

since $p_1\langle a \rightsquigarrow at(p_2)\rangle$ is ndi in p_2 and by the induction hypothesis, and

$$(**) \leq \tfrac{1}{2} d(F_1, F_2)$$

since $p_1\langle a \rightsquigarrow at(p_2)\rangle$ is ndi in p_1 and by the induction hypothesis. □