Denotational Semantics for Unguarded Recursion: The Demonic Case

A. de Bruin

Faculty of Economics, Erasmus Universiteit
P.O. Box 1738, NL-3000 DR Rotterdam, arie@cs.eur.nl

E.P. de Vink

Department of Mathematics and Computer Science, Vrije Universiteit
De Boelelaan 1081, NL-1081 HV Amsterdam, vink@cs.vu.nl

ABSTRACT We show that the technique to prove equivalence of operational and denotational cpo based semantics using retractions, as introduced in [BV] for a sequential backtracking language, can be applied to parallel languages as well. We prove equivalence for a uniform language in which procedure calls need not be guarded. The unguardedness is taken care of by giving a semantics in which the nondeterminism is demonic.

Section 1 Introduction

In [BV]a we have introduced a new idea of proving equivalence of an operational semantics, say $\mathcal{O}$, and a denotational semantics, say $\mathcal{D}$, where both semantics are based on cpo's. The idea is to introduce less abstract variants $\tilde{\mathcal{O}}$ and $\tilde{\mathcal{D}}$ of the semantical definitions for which the equivalence $\tilde{\mathcal{O}} = \tilde{\mathcal{D}}$ is easier to establish. The latter proof will be less complicated if $\tilde{\mathcal{O}}$ and $\tilde{\mathcal{D}}$ can be defined in such a way that for each statement they deliver maximal elements in the cpo of meanings of statements. The equivalence $\mathcal{O} = \mathcal{D}$ can then be established by showing that we have both $\mathcal{O} = \rho(\tilde{\mathcal{O}})$ and $\mathcal{D} = \rho(\tilde{\mathcal{D}})$ for a suitable abstraction operator ρ. The idea was tested against an operational and denotational model for a sequential backtracking language.

In this paper we will apply this technique to a rather simple but parallel language, featuring atomic actions a, procedure calls x, sequential composition $\cdot$, nondeterministic choice $+$ and parallel merge $\|$. The language is uniform: the meaning of a statement will be a set of streams, where streams are sequences of atomic actions. (Uniform taken in the sense of [BKMOZ], streams taken in the sense of [Me].) A subclass of the class of all such sets can be turned into a cpo using the Egli-Milner ordering (as in [Pl]). Such a powerdomain can be manipulated using techniques as developed in [MV]. (Strictly speaking, in [MV]a the Smyth ordering of [Sm] is used, however the results developed there also apply for the Egli-Milner ordering.)

The complicating element in the language however is that we allow for unguarded procedure calls. This means that it is not guaranteed that in between two successive procedure calls at least one atomic action is performed, and this can lead to infinite chains of procedure calls which are not visible in the resulting streams. For instance, let procedure x be defined with body $x \cdot a + b$. Intuitively

speaking, the meaning of a call of x should be the set $\{ ba^n \mid n \geq 0 \} \cup \{\bot\}$, where the result $\bot$ models the internal divergence due to an infinite chain of calls of x.

A proper treatment giving full account to internal divergence in the cpo framework has, to our knowledge, not been established yet. In this paper we will adopt a standard way out, cf. [BHR], [Me]a, by modeling demonic nondeterminism: if executing a statement s leads, after a few atomic actions $a_1, \cdot\cdot, a_n$, to an infinite chain of procedure calls, then our semantic operators $\mathcal{O}$ and $\mathcal{D}$ applied to s deliver a set of streams which contains the corresponding stream $a_1 \cdot\cdot a_n\bot$, but no other streams of the form $a_1 \cdot\cdot a_n x'$, even if such streams would correspond to other computations of s. In the example above this means that the meaning of the call x will be "flattened out" into the singleton set $\{\bot\}$.

After introducing a few mathematical tools in section 2, we will devote sections 3 and 4 to a definition of the operational and the denotational semantics of our language by introducing the semantical mappings $\mathcal{O}$ and $\mathcal{D}$, respectively. Section 5 will then be dedicated to the equivalence proof of $\mathcal{O}$ and $\mathcal{D}$. This proof will proceed along the lines sketched in the beginning of this introduction. In order to appreciate the proof it is first of all relevant to observe that, if we would allow only guarded procedure calls in our language, this proof would not be very complicated. For in that case one can prove that both semantic operators assign to each statement a meaning that is a maximal element in the powerdomain based on the Egli-Milner ordering described above. Equivalence can then, for instance, be proved using a characterization of $\mathcal{O}$ as the unique fixed point of a higher order transformation. Cf. [KR] for this approach in a metrical setting.

However to deal with the present situation first we will introduce in section 5 the variants $\tilde{\mathcal{O}}$ and $\tilde{\mathcal{D}}$ of $\mathcal{O}$ and $\mathcal{D}$, which differ from the latter semantics only in the fact that for each execution of a procedure call a "silent step" τ is recorded in the resulting stream. These new semantical models would deliver as the meaning of the procedure x, discussed above, the maximal set $\{ \tau^n ba^n \mid n \geq 0 \} \cup \{ \tau^\omega \}$. One could say that the effect of these new semantical operators is that, in an artificial way, all procedure calls have been made guarded. (An idea also present in [BBKM].) The equivalence of $\tilde{\mathcal{O}}$ and $\tilde{\mathcal{D}}$ will then be established by the standard metrical technique but now in the setting of cpo's.

The next step to take is to define an abstraction operator ρ for which the equalities $\mathcal{O} = \rho(\tilde{\mathcal{O}})$ and $\mathcal{D} = \rho(\tilde{\mathcal{D}})$ should hold. This abstraction operator will be derived from an abstraction operator with the same name, which operates on streams by removing all τ's. For instance we would have that $\rho(\tau\tau baa) = baa$, and that $\rho(\tau^\omega) = \bot$. The set $\mathcal{A}^{st}_\tau$ of all streams built up from atomic actions and τ's is a cpo. This also holds for the set $\mathcal{A}^{st}$ of the streams containing only atomic actions. These two cpo's are related through the operator ρ. This operator can be extended to the powerdomains built upon $\mathcal{A}^{st}_\tau$ and $\mathcal{A}^{st}$. In [BV]a we have developed a little theory on such pairs of cpo's related through an abstraction operator like ρ. This theory will be used to derive the result that the operators $\mathcal{O}$ and $\mathcal{D}$, defined on the abstract versions of the cpo's, can be viewed as the abstractions of the operators $\tilde{\mathcal{O}}$ and $\tilde{\mathcal{D}}$ defined on their less abstract counterparts. More specifically, we can extend the abstraction operator ρ to functions on the cpo's occurring in this paper, and prove the desired result $\mathcal{O} = \rho(\tilde{\mathcal{O}})$ and $\mathcal{D} = \rho(\tilde{\mathcal{D}})$. This will complete the equivalence proof $\mathcal{O} = \mathcal{D}$.

Although this result is interesting in its own right, it can also serve as a preparatory step towards establishing a semantics of parallelism with unguarded procedure calls, that is more intuitive than the demonic version presented in this paper, i.e., that captures the full meaning of unguarded calls instead

of the subset provided by the demonic semantical operators where denotationally the presence of divergence, i.e. $\perp$, makes other outcomes invisible. The problem is that it is not clear how to define a *cpo* consisting of meanings of such programs, e.g., the set $\{ ba^n \mid n \geq 0 \} \cup \{\perp\}$, discussed earlier. On the other hand, it seems possible to extend the theory from [BV]a such that only the less abstract domains should be cpo's. For the abstract versions of the domains, for instance in our case the power-domain based on $\mathcal{A}^{st}$, we do not need the full structure of a cpo, an ordering that need not even be anti-symmetric seems to be sufficient. We are investigating whether this observation can be made to work.

Acknowledgements. The results in this paper have been presented in a meeting of the CWI working group on the semantics of concurrency, leading to a fruitful discussion. It is a pleasure to thank the members of this group. Moreover, as always, we are indebted to M279 for her hospitality.

Section 2 Mathematical Preliminaries

In this section we present the mathematical prerequisites for the construction and equivalence of the semantical definitions in this paper. To start with we introduce an Egli-Milner powerdomain of streams similar to the one that was introduced in [MV]a with respect to the Smyth ordering.

(2.1) DEFINITION

(i) Let $\mathcal{A}$ be an alphabet. Distinguish $\perp \notin \mathcal{A}$. The set of streams $\mathcal{A}^{st}$ over $\mathcal{A}$ is given by $\mathcal{A}^{st} = \mathcal{A}^* \cup \mathcal{A}^* \cdot \perp \cup \mathcal{A}^\omega$. For $x \in \mathcal{A}^{st}$ and $n \in \mathbb{N}$ we define $x[n] \in \mathcal{A}^{st}$ as follows: $x[0] = \perp$, $\perp[n+1] = \perp$, $\varepsilon[n+1] = \varepsilon$ and $(ax')[n+1] = a(x'[n])$. We stipulate $x[\infty] = x$. The stream ordering $\leq_{st}$ on $\mathcal{A}^{st}$ is defined as follows: $x \leq_{st} y \Leftrightarrow \exists \alpha \in \mathbb{N}_\infty : x = y[\alpha]$, with $\mathbb{N}_\infty = \mathbb{N} \cup \{\infty\}$.

(ii) Let $X \subseteq \mathcal{A}^{st}$. X is *flat* $\Leftrightarrow \forall x, x' \in X: \neg (x <_{st} x')$. $X[n] = \{ x[n] \mid x \in X \}$. X is *closed* $\Leftrightarrow \forall x \in \mathcal{A}^{st}: (\forall n \in \mathbb{N}: x[n] \in X[n]) \Rightarrow x \in X$. X is *bounded* $\Leftrightarrow \forall n \in \mathbb{N}: X[n]$ is finite. X is *compact* $\Leftrightarrow X$ is flat, closed and bounded.

(iii) Let $\mathcal{P}^*(\mathcal{A}^{st})$ denote the collection of all compact and non-empty subsets of $\mathcal{A}^{st}$. The Egli-Milner ordering $\leq_{EM}$ on $\mathcal{P}^*(\mathcal{A}^{st})$ is defined by $X \leq_{EM} Y \Leftrightarrow \forall x \in X \exists y \in Y: x \leq_{st} y \ \& \ \forall y \in Y \exists x \in X: x \leq_{st} y$.

Streams are finite or infinite sequences of (abstract) actions, possibly ending in a distinguished marker $\perp$ indicating that this stream is not completed (yet). In order to obtain a suitable powerdomain we have to impose a restriction on the sets of streams we allow in the domain $\mathcal{P}^*(\mathcal{A}^{st})$. Flatness is needed for the anti-symmetry of the Egli-Milner ordering; closedness and boundedness are needed to support the lifting lemma below. Non-emptiness is just for convenience, for the bottom element $\{\perp\}$ does not compare immediately to $\emptyset$.

For a stream set X we use $min(X)$ to denote its subset of minimal elements. For a mapping $f: (\mathcal{A}^{st})^k \to \mathcal{P}^*(\mathcal{A}^{st})$ we use $f[\cdot]$ to denote the induced mapping on $(\mathcal{P}(\mathcal{A}^{st}))^k$, thus $f[\vec{X}] = \cup \{ f(\vec{x}) \mid \vec{x} \in \vec{X} \}$ (using an obvious vector notation).

The main virtue of using compact stream sets is the availability of a lifting lemma. In

combination with an extension lemma this provides a convenient tool for the definition of continuous operators. See [Vi2] for some applications of this technique.

(2.2) THEOREM

(i) $(\mathcal{A}^{st}, \leq_{st})$ and $(\mathcal{P}^*(\mathcal{A}^{st}), \leq_{EM})$ are cpo's.

(ii) (Extension Lemma) Put $\mathcal{A}^f = \mathcal{A}^* \cup \mathcal{A}^* \cdot \perp$. Let D be an cpo. Suppose $f : (\mathcal{A}^f)^k \to D$ is monotonic. Then $\overline{f} : (\mathcal{A}^{st})^k \to D$ defined by $\overline{f}(\vec{x}) = lub_i\ f(\vec{x}[i])$ is well-defined and continuous. $\overline{f}$ is called the extension of f.

(iii) (Lifting Lemma) If $f : (\mathcal{A}^{st})^k \to \mathcal{P}^*(\mathcal{B}^{st})$ is continuous and $F : (\mathcal{P}^*(\mathcal{A}^{st}))^k \to \mathcal{P}^*(\mathcal{B}^{st})$ is given by $F(\vec{X}) = min(\ f[\vec{X}]\ )$ then F is well-defined and continuous. F is called the lifted version of f.

PROOF Adaptation to the Egli-Milner framework of the proofs in [MV]a. □

The equivalence proof in section 5 intuitively makes a distinction between abstract and concrete cpo's. This intuition is captured by the notion of retraction as was introduced in [BV]a and repeated here. The associated notion of a canonical mapping is presented here as well, together with a lemma related to lifting (as introduced above). We conclude this section with a useful lemma relating the least fixed point of a mapping to the least fixed point of its retract.

(2.3) DEFINITION

(i) Let $D, \tilde{D}$ be cpo's. D is called a retract of $\tilde{D}$ if there exist two continuous mappings $i : D \to \tilde{D}$, $j : \tilde{D} \to D$ such that $j \circ i = id_D$. Notation: $D \leq_{i,j} \tilde{D}$.

(ii) Suppose $D \leq_{i,j} \tilde{D}$, $E \leq_{k,l} \tilde{E}$. A mapping $\tilde{\phi} : \tilde{D} \to \tilde{E}$ is called canonical if there exists $\phi : D \to E$ such that $l \circ \tilde{\phi} = \phi \circ j$. The function space $\tilde{D}$ ~> $\tilde{E}$ of canonical mappings from $\tilde{D}$ to $\tilde{E}$ is given by $\tilde{D}$ ~> $\tilde{E}$ = { $\tilde{\phi} : \tilde{D} \to \tilde{E}$ | $\tilde{\phi}$ canonical }.

Note that if $\tilde{\phi}$ is canonical, there exist a unique ϕ satisfying $l \circ \tilde{\phi} = \phi \circ j$. This ϕ is called the retract of $\tilde{\phi}$ (with respect to $D \leq_{i,j} \tilde{D}$ and $E \leq_{k,l} \tilde{E}$).

If D is a retract of $\tilde{D}$, say $D \leq_{i,j} \tilde{D}$, then we have an equivalence relation $\sim_D$ on $\tilde{D}$ (induced by j) defined by $d \sim_D d' \Leftrightarrow j(d) = j(d')$. For $\tilde{\phi} : \tilde{D} \to \tilde{E}$ we can reformulate canonicity in terms of the equivalence relations $\sim_D$ and $\sim_E$: $\tilde{\phi} : \tilde{D} \to \tilde{E}$ is canonical iff the induced mapping on equivalence classes $\tilde{\phi} : \tilde{D} / \sim_D \to \tilde{E} / \sim_E$ is well defined. Therefore it suffices to check that equivalent arguments map onto equivalent images for a function to be canonical.

Suppose $D \leq_{i,j} \tilde{D}$ and $E \leq_{k,l} \tilde{E}$. The function space $D \to E$ then, will be a retract of the function space $\tilde{D}$ ~> $\tilde{E}$. For, if we define $I = \lambda\phi\,.\,k \circ \phi \circ j$ and $J = \lambda\tilde{\phi}\,.\,l \circ \tilde{\phi} \circ i$ then we have $(\ D \to E\ ) \leq_{I,J} (\ \tilde{D}$ ~> $\tilde{E}\ )$. Note that $J(\tilde{\phi})$ is the retract of $\tilde{\phi}$. Analogously, if V is a set of values and $D \leq_{i,j} \tilde{D}$ then $V \to D \leq_{I,J} V \to \tilde{D}$ where $I = \lambda\phi\,.\,i \circ \phi$ and $J = \lambda\tilde{\phi}\,.\,j \circ \tilde{\phi}$.

Next we check that the powerdomain construction $\mathcal{P}^*(\cdot)$ respects, so to speak, the retraction ordering. If $\mathcal{A}^{st} \leq_{i,j} \tilde{\mathcal{A}}^{st}$ then the induced pair of functions I, J shows that the domain $\mathcal{P}^*(\mathcal{A}^{st})$ is a retract of $\mathcal{P}^*(\tilde{A}^{st})$.

(2.4) LEMMA *Assume* $\mathcal{A}^{st} \leq_{i,j} \tilde{\mathcal{A}}^{st}$. *Let I and J be the lifted versions of* $\lambda x.\{\, i(x) \,\}$ *and* $\lambda \tilde{x}.\{\, j(\tilde{x}) \,\}$, *respectively. Then it holds that* $\mathcal{P}^*(\mathcal{A}^{st}) \leq_{I,J} \mathcal{P}^*(\tilde{A}^{st})$.

PROOF Let $X \in \mathcal{P}^*(\mathcal{A}^{st})$. We have to show that $J(I(X)) = X$. Now $J(I(X)) = min(\, j[\, min(\, i[X]\,)\,]\,)$ (by definition) $= min(\, j[\, i[X]\,]\,)$ (by monotonicity of j) $= min(X)$ (since $j \circ i = id_{\mathcal{A}^{st}}$) $= X$ (since X is flat). □

The next lemma gives a sufficient condition for two lifted mappings being related canonically, thus combining our two notions of lifting and retraction.

(2.5) LEMMA *Assume* $\mathcal{A}^{st} \leq_{i,j} \tilde{\mathcal{A}}^{st}$. *Suppose* f: $(\mathcal{A}^{st})^k \to \mathcal{P}^*(\mathcal{A}^{st})$ *is monotonic and* $\tilde{f}$: $(\tilde{\mathcal{A}}^{st})^k \to \mathcal{P}^*(\tilde{A}^{st})$ *is such that* $J \circ \tilde{f} = f \circ j$. *Let* F: $(\mathcal{P}^*(\mathcal{A}^{st}))^k \to \mathcal{P}^*(\mathcal{A}^{st})$, $\tilde{F}$: $(\mathcal{P}^*(\tilde{A}^{st}))^k \to \mathcal{P}^*(\tilde{A}^{st})$ *be the lifted versions of* f *and* $\tilde{f}$, *respectively. Then it holds that* $\tilde{F}$ *is canonical with retract* F.

PROOF We have to show that $J \circ \tilde{F} = F \circ J$. Pick $\tilde{X} \in (\mathcal{P}^*(\tilde{A}^{st}))^k$. Then

$J(\tilde{F}(\tilde{X})$
$= min(\, j[\, min(\, \tilde{f}[\tilde{X}]\,)\,]\,)$ (by definition of J and $\tilde{F}$)
$= min(\, j[\, \tilde{f}[\tilde{X}]\,]\,)$ (by monotonicity of $j[\cdot]$)
$= min(\, f[\, j[\tilde{X}]\,]\,)$ (by the assumption)
$= min(\, f[\, min(\, j[\tilde{X}]\,)\,]\,)$ (by monotonicity of f)
$= F(J(\tilde{X}))$ (by definition of F and J). □

Next we state a theorem taken from [BV]a that we will use to relate concrete and abstract fixed points in the equivalence proof of the operational and denotational models below. It is similar to the Fixed Point Transformation Lemma of [BMZ], [Me]a.

(2.6) THEOREM *Suppose* $D \leq_{i,j} \tilde{D}$. *Let* $\tilde{\Phi} : \tilde{D} \to \tilde{D}$ *be continuous and canonical with retract* Φ. *Then* $\Phi : D \to D$ *is continuous with* $\mu\Phi = j(\mu\tilde{\Phi})$. □

The domains on which we base the semantical definition in the remaining sections are built up from the stream domains $\mathcal{A}^{st}$ and $\mathcal{A}_\tau^{st}$ (where $\mathcal{A}_\tau = \mathcal{A}_\tau \cup \{\tau\}$ for some special symbol τ; see definition 3.2 below). The inclusion mapping ι and the mapping ρ: $\mathcal{A}_\tau^{st} \to \mathcal{A}^{st}$ to be given in a minute show that the domain $\mathcal{A}^{st}$ is a retract of $\mathcal{A}_\tau^{st}$. Intuitively, the map ρ is a τ-remover, replacing finitely many juxtaposed τ's by ε but infinitely many by $\perp$.

(2.7) DEFINITION Let ι: $\mathcal{A}^{st} \to \mathcal{A}_\tau^{st}$ be the inclusion mapping. Define ρ: $\mathcal{A}_\tau^{st} \to \mathcal{A}^{st}$ as the unique continuous mapping such that

$\rho(\perp) = \perp$, $\rho(\varepsilon) = \varepsilon$, $\rho(ax) = a\rho(x)$ and $\rho(\tau x) = \rho(x)$.

Let I and P denote the lifted versions of $\lambda x.\{\, x \,\}$ and $\lambda x.\{\, \rho(x) \,\}$ on $\mathcal{P}^*(\mathcal{A}^{st})$ and $\mathcal{P}^*(\mathcal{A}_\tau^{st})$, respectively.

We leave it to the reader to check that $\mathcal{A}^{st}$ is indeed a retract of $\mathcal{A}_\tau^{st}$. (This can be established straightforwardly using stream induction.)

(2.8) LEMMA $\mathcal{A}^{st} \leq_{\iota,\rho} \mathcal{A}_\tau^{st}$ *and* $\mathcal{P}^*(\mathcal{A}^{st}) \leq_{I,P} \mathcal{P}^*(\mathcal{A}_\tau^{st})$. □

In the sequel we will suppress ι (and I) but use ρ also for mappings on powerdomains, products and function spaces induced by ρ on $\mathcal{A}_\tau^{st}$, as we expect no confusion to arise.

Section 3 The Language and Its Operational Semantics

In this section we introduce the simple programming language *Prog* for which we shall illustrate our method of designing equivalent operational and denotational semantics. We provide a Plotkin style (parametrized) transition system (cf. [HP]) that represents the computation steps of an abstract machine running the language. From this we extract the operational model for *Prog*.

(3.1) DEFINITION Fix a set $\mathcal{A}$ and a set $\mathcal{X}$, the elements of which are called actions and procedure variables, respectively. $\mathcal{A}$ is ranged over by a, $\mathcal{X}$ is ranged over by x. The class of statements *Stat*, with typical element s, is given by

$$s ::= a \mid x \mid s_1 \cdot s_2 \mid s_1 + s_2 \mid s_1 \| s_2.$$

The class of declarations *Decl*, ranged over by d, has elements of the format $x_1 \Leftarrow s_1 : \cdots : x_n \Leftarrow s_n$ where $n \in \mathbb{N}$, $x_i \in \mathcal{X}$ all distinct and $s_i \in Stat$, $(i \in \{1, .., n\})$. The class of programs *Prog* consists of pairs $d \mid s$, where $d \in Decl$ and $s \in Stat$.

A statement is either an action a in $\mathcal{A}$, a procedure call x in $\mathcal{X}$, a sequential composition $s_1 \cdot s_2$, a nondeterministic choice $s_1 + s_2$ or a concurrent execution $s_1 \| s_2$.

Our main interest here is in the programming concepts embodied by the syntactical operators and by recursion. Therefore, the programming language *Prog* under consideration is kept uniform or schematic, since the elementary actions are left unspecified. (See [BKMOZ]a for a discussion on uniform vs. non-uniform semantics.)

Let $d = x_1 \Leftarrow s_1 : \cdots : x_n \Leftarrow s_n$ be a declaration in *Decl*. Define $d(x) = s_i$ if $x = x_i$ for some $i \in \{1, .., n\}$, and $d(x) = x$ otherwise. Note that by this convention a procedure variable is always declared, since by default it has itself as its body.

We will base the operational semantics of a program $d \mid s$ on a labeled transition system indexed by d to be given in a moment. First we need an auxiliary definition.

(3.2) DEFINITION Distinguish $E \notin Stat$. E is called the empty statement. Define the class of generalized statements $Stat_E$, with typical element $\bar{s}$, by $Stat_E = Stat \cup \{E\}$. Distinguish moreover $\tau \notin \mathcal{A}$. τ is called the silent step. The class of generalized actions $\mathcal{A}_\tau$, ranged over by α, is given by $\mathcal{A}_\tau = \mathcal{A} \cup \{\tau\}$.

The empty statement E is associated with successful termination, (cf. [Ap]). It will be convenient below to allow the expressions $\bar{s} \cdot \bar{s}'$, $\bar{s} + \bar{s}'$ and $\bar{s} \parallel \bar{s}'$ for arbitrary $\bar{s}, \bar{s}' \in \mathit{Stat}_{\mathrm{E}}$. Therefore we stipulate $\bar{s} * \mathrm{E} = \mathrm{E} * \bar{s} = \bar{s}$ for $\bar{s} \in \mathit{Stat}_{\mathrm{E}}$, $* \in \{\cdot, +, \parallel\}$.

The silent step τ - which plays a predominant role in algebraic approaches to the semantics of concurrency as, e.g., [Mi], [BK] - will be used in definition 3.3 below to indicate body replacement. Thus we can model procedure calls quite naturally using an axiom instead of a rule. (Similar to [BMOZ] for the μ-construct.) In the operational semantics $\mathcal{O}$ we abstract from this τ's, but we will again take them into account with respect to a modified operational model $\tilde{\mathcal{O}}$ in section 5.

(3.3) DEFINITION A declaration $d \in \mathit{Decl}$ induces a labeled transition system d which is the smallest subset of $\mathit{Stat}_{\mathrm{E}} \times \mathcal{A}_\tau \times \mathit{Stat}_{\mathrm{E}}$ such that (using an obvious infix notation) it holds that

$$a \to_d^a \mathrm{E} \qquad \text{(Action)}$$

$$x \to_d^\tau s \quad \textit{where } s = d(x) \qquad \text{(Proc)}$$

$$\frac{s \to_d^\alpha \bar{s}}{s \cdot s' \to_d^\alpha \bar{s} \cdot s'} \qquad \text{(Seq)}$$

$$\frac{s \to_d^\alpha \bar{s}}{s + s' \to_d^\alpha \bar{s} \quad s' + s \to_d^\alpha \bar{s}} \qquad \text{(Choice)}$$

$$\frac{s \to_d^\alpha \bar{s}}{s \parallel s' \to_d^\alpha \bar{s} \parallel s' \quad s' \parallel s \to_d^\alpha s' \parallel \bar{s}} \qquad \text{(Par)}$$

The axiom (Action) states that an action $a \in \mathcal{A}$ can always terminate successfully after performing a. If a procedure call x is about to be executed, a silent step τ is signaled and the computation continues with the body s of x, which is looked up in the declaration. The three rules (Seq), (Choice) and (Par) can be used to unravel composite statements. Alternative and parallel compositions can perform the same actions (in $\mathcal{A}_\tau$) as their constituting components do. For a sequential composition $s \cdot s'$, however, this depends on the actions of its first component, for intuitively the execution of s' is started after the execution of s has finished.

(3.4) DEFINITION Define the operational semantics $\mathcal{O}: \mathit{Prog} \to \mathcal{P}(\mathcal{A}^{st})$ as follows: $x \in \mathcal{O}(d \mid s)$ iff either of the two following cases holds:

(1) $\exists n, \exists \bar{s}_0, \cdots, \bar{s}_n, \exists \alpha_1, \cdots, \alpha_n : \bar{s}_0 = s, \bar{s}_{i-1} \to_d^{\alpha_i} \bar{s}_i$ for $\forall i \in \{1, .., n\}, \bar{s}_n = \mathrm{E} \;\&\; x = \rho(\alpha_1 \cdots \alpha_n)$;

(2) $\exists \bar{s}_0, \bar{s}_1, \cdots, \exists \alpha_1, \alpha_2, \cdots : \bar{s}_0 = s, \bar{s}_{i-1} \to_d^{\alpha_i} \bar{s}_i$ for $\forall i \in \{1, 2, ...\} \;\&\; x = \rho(\alpha_1 \alpha_2 \cdots)$.

For a program $d \mid s$ the operational semantics collects all the strings of labels of (i) terminating and (ii) infinite d-computations starting from s. The τ's in definition 3.3. are needed to treat infinite behavior. However, note that by application of the τ-remover ρ we abstract from steps due to procedure calls. Internal divergence as, e.g., for the program $x \Leftarrow x \mid x$ is represented by $\perp$, since $\rho(\tau^\omega)$

equals $\perp$. (In the equivalence proof to be presented in the sequel however we will leave τ^ω as it stands, thus making internal divergence observable.)

EXAMPLE Let d denote the declaration $x \Leftarrow x \cdot a + b$. Then the operational semantics of $d \mid x$ does not only contain ba^* but also $\perp$, since there exists a computation which generates solely τ-labels by always choosing the left alternative. So $\mathcal{O}(d \mid x) = \{\perp\} \cup ba^*$.

Section 4 The Denotational Semantics for *Prog*

In this section we present a denotational semantics for the language under consideration. It will assign to each program a *denotation*, i.e. an object representing its meaning, in a suitably chosen mathematical domain, and moreover the semantics will be compositional or homomorphic: the meanings of a composite construct depends only upon the meaning of its constituting components.

Programs will be given meaning in the domain $\mathcal{P}^*(\mathcal{A}^{st})$. So compact stream sets serve as denotations. In order to deal with procedure calls we introduce the notion of an environment. Environments are used to store and retrieve the meaning of procedure variables, i.e., they are mappings from $\mathcal{X}$ to $\mathcal{P}^*(\mathcal{A}^{st})$. For a program $d \mid s$, the statement s will be evaluated with respect to an environment depending on the declaration d. So defining the denotational semantics amounts to specifying the evaluation of a statement and the environment corresponding to a given declaration.

The compositionality requirement for the evaluation of statements will be met by designing for each syntactical operator, viz. $\cdot$, $+$ and $\|$, a semantical one, also written as $\cdot$, $+$ and $\|$, respectively. We will have $\mathcal{D}(d \mid s_1 * s_2) = \mathcal{D}(d \mid s_1) * \mathcal{D}(d \mid s_2)$ for each $* \in \{\cdot, +, \|\}$. Using the definition of the statement evaluator $\mathcal{S}$ we derive for each declaration an environment transformation. We will take the least fixed point, say η_d, of this transformation, for the environment associated with the declaration. For this to work, i.e., for η_d to exist, we need that this transformation is continuous. Since the transformation of an environment η is in essence body replacement - a procedure variable will be mapped on the denotation of its body with respect to η - we will have to assure continuity of the semantical operators $\cdot$, $+$ and $\|$. In its turn this will be guaranteed by the extension and lifting lemma, presented in section 2, that we will use in the construction of these operators.

(4.1) DEFINITION Define $\cdot$ on $\mathcal{P}^*(\mathcal{A}^{st})$ as the extended and lifted version of $\cdot : \mathcal{A}^f \times \mathcal{A}^f \to \mathcal{P}^*(\mathcal{A}^{st})$ given by

$\varepsilon \cdot y = \{y\}$, $\perp \cdot y = \{\perp\}$ and $\alpha x \cdot y = \alpha(x \cdot y)$.

Define $+$ on $\mathcal{P}^*(\mathcal{A}^{st})$ as the extended and lifted version of $+ : \mathcal{A}^f \times \mathcal{A}^f \to \mathcal{P}^*(\mathcal{A}^{st})$ given by

$x + y = min\{x, y\}$.

Let $\|$, $\lfloor\!\lfloor : \mathcal{A}^f \times \mathcal{A}^f \to \mathcal{P}^*(\mathcal{A}^{st})$ be given by

$x \| y = x \lfloor\!\lfloor y + y \lfloor\!\lfloor x$,

$\varepsilon \lfloor\!\lfloor y = \{y\}$, $\perp \lfloor\!\lfloor y = \{\perp\}$ and $\alpha x \lfloor\!\lfloor y = \alpha(x \| y)$.

Define $\|$ on $\mathcal{P}^*(\mathcal{A}^{st})$ as the extended and lifted version of $\|$ on $\mathcal{A}^f$.

Having now available semantical interpretations for our syntactical operators we can proceed

with the definition of the denotational semantics for *Prog*. It follows the same general scheme as in, e.g., [BM], [Vi1].

(4.2) DEFINITION Define the collection of environments *Env*, with typical element η, by $Env = \mathcal{X} \to \mathcal{P}^*(\mathcal{A}^{st})$. Define the statement evaluator $\mathcal{S}$: $Stat \to Env \to \mathcal{P}^*(\mathcal{A}^{st})$ by

$\mathcal{S}(a)(\eta) = \{a\}$,

$\mathcal{S}(x)(\eta) = \eta(x)$,

$\mathcal{S}(s_1 * s_2)(\eta) = \mathcal{S}(s_1)(\eta) * \mathcal{S}(s_2)(\eta)$ for $* \in \{\cdot, +, \|\}$.

Let for $d \in Decl$ the environment transformation H_d: $Env \to Env$ be given by

$H_d(\eta)(x) = \mathcal{S}(s)(\eta)$

where $s = d(x)$. The denotational semantics $\mathcal{D}$: $Prog \to \mathcal{P}^*(\mathcal{A}^{st})$ is defined by

$\mathcal{D}(d \mid s) = \mathcal{S}(s)(\eta_d)$

where η_d is the least fixed point of H_d.

We leave it to the reader to check the well-definedness of the semantical mapping $\mathcal{D}$. (Consult, e.g., [Ba].) In the next section we will relate this denotational definition with the operational model of the previous one.

EXAMPLE Consider again the program $d \mid x$ where $d = x \Leftarrow x \cdot a + b$. Let $\eta_0 = \lambda x.\{\bot\}$ be the least environment in *Env*. Then we have $H_d(\eta_0)(x) = \mathcal{S}(x \cdot a + b)(\eta_0) = \{\bot\} \cdot \{a\} + \{b\} = \{\bot\} + \{b\} = \{\bot\}$. Therefore iteration of H_d gives a least fixed point η_d with $\eta_d(x) = \{\bot\}$. Thus $\mathcal{D}(d \mid x) = \{\bot\}$. So the other outcomes, ba^* as present in the operational meaning of $d \mid x$ have vanished.

Section 5 Equivalence of $\mathcal{O}$ and $\mathcal{D}$

In this section we will compare the two models $\mathcal{O}$ and $\mathcal{D}$ using our little theory of retracts. We will introduce less abstract semantical definitions $\tilde{\mathcal{O}}_d$ and $\tilde{\mathcal{D}}_d$ (for $d \in Decl$) in which the silent τ-steps can be observed. We relate the abstract and concrete models for the operational and denotational case. The picture is then completed by establishing $\tilde{\mathcal{O}}_d = \tilde{\mathcal{D}}_d$ in a [KR]a-style, using a higher-order transformation with a unique fixed point.

We start with defining a τ-version of the operational semantics and relate it to the original one.

(5.1) DEFINITION Let $d \in Decl$. Define the semantical mapping $\tilde{\mathcal{O}}_d$: $Stat_E \to \mathcal{P}^*(\mathcal{A}^{st}_\tau)$ by $\tilde{\mathcal{O}}_d(\bar{s}) = min\{\, x \in \mathcal{A}^{st}_\tau \mid \exists n,\ \exists \bar{s}_0, .., \bar{s}_n,\ \exists \alpha_1, .., \alpha_n\colon\ \bar{s}_0 = \bar{s},\ \bar{s}_{i-1} \to^{\alpha_i}_d \bar{s}_i,\ \bar{s}_n = E,\ x = \alpha_1 \cdot\cdot \alpha_n$ or $\exists \bar{s}_0, \bar{s}_1, \ldots \exists \alpha_1, \alpha_2, \ldots : \bar{s}_0 = \bar{s}, \bar{s}_{i-1} \to^{\alpha_i}_d \bar{s}_i, x = \alpha_1 \alpha_2 \cdots \}$.

$\tilde{\mathcal{O}}_d$ is well-defined; the closedness of $\tilde{\mathcal{O}}_d(s)$ for $s \in Stat_E$ follows by König's lemma. The abstraction ρ has disappeared when compared to definition 3.4. Note the application of the *min*-operator in the above definition. Because of this (and of the implicit *min* in the lifted ρ on stream

sets) we do not have $\rho(\tilde{\mathcal{O}}_d(s)) = \mathcal{O}(d \mid s)$ since the operational definition $\mathcal{O}$ does not necessarily deliver flat sets.

(5.2) LEMMA *For $d \in Decl$ and $s \in Stat$ it holds that* $\rho(\tilde{\mathcal{O}}_d(s)) = min(\mathcal{O}(d \mid s))$.

PROOF Let $d \mid s \in Prog$. From the definitions of $\mathcal{O}$ and $\tilde{\mathcal{O}}_d$ we see

$\rho(\tilde{\mathcal{O}}_d(s))$

$= min(\rho[\tilde{\mathcal{O}}_d(s)])$ (by definition of ρ on $\mathcal{P}^*(\mathcal{A}_\tau^{st})$)

$= min(\rho[\, min\{\, x \mid s = \bar{s}_0 \to_d^{\alpha_1} \cdots \to_d^{\alpha_n} \bar{s}_n = \mathrm{E},\ x = \alpha_1 \cdot\cdot \alpha_n \text{ or } s = \bar{s}_0 \to_d^{\alpha_1} s_1 \to_d^{\alpha_2} \cdots,$
$x = \alpha_1\alpha_2 \cdots \,\}\,])$ (by definition of $\tilde{\mathcal{O}}_d$)

$= min(\rho[\, \{\, x \mid s = \bar{s}_0 \to_d^{\alpha_1} \cdots \to_d^{\alpha_n} \bar{s}_n = \mathrm{E},\ x = \alpha_1 \cdot\cdot \alpha_n \text{ or } s = \bar{s}_0 \to_d^{\alpha_1} s_1 \to_d^{\alpha_2} \cdots,$
$x = \alpha_1\alpha_2 \cdots \,\}\,])$ (by monotonicity of ρ)

$= min\ \{\, x \mid s = \bar{s}_0 \to_d^{\alpha_1} \cdots \to_d^{\alpha_n} \bar{s}_n = \mathrm{E},\ x = \rho(\alpha_1 \cdot\cdot \alpha_n) \text{ or } s = \bar{s}_0 \to_d^{\alpha_1} s_1 \to_d^{\alpha_2} \cdots,$
$x = \rho(\alpha_1\alpha_2 \cdots)\,\}$ (by definition of $\rho[\cdot]$)

$= min(\mathcal{O}(d \mid s))$. □

We continue with the construction of a concrete counterpart of $\mathcal{D}$. For this we first extend the semantical operators to the domain $\mathcal{P}^*(\mathcal{A}_\tau^{st})$.

(5.3) DEFINITION Define $\tilde{\cdot}$ on $\mathcal{P}^*(\mathcal{A}_\tau^{st})$ as the extended and lifted version of $\tilde{\cdot}: \mathcal{A}_\tau^f \times \mathcal{A}_\tau^f \to \mathcal{P}^*(\mathcal{A}_\tau^{st})$ given by

$\varepsilon \,\tilde{\cdot}\, y = \{y\}$, $\perp \,\tilde{\cdot}\, y = \{\perp\}$ and $\alpha x \,\tilde{\cdot}\, y = \alpha(x \,\tilde{\cdot}\, y)$.

Define $\tilde{+}$ on $\mathcal{P}^*(\mathcal{A}_\tau^{st})$ as the extended and lifted version of $\tilde{+}: \mathcal{A}_\tau^f \times \mathcal{A}_\tau^f \to \mathcal{P}^*(\mathcal{A}_\tau^{st})$ given by

$x \,\tilde{+}\, y = min\{\, x, y \,\}$.

Let $\tilde{\|}$, $\tilde{\lfloor\!\lfloor}: \mathcal{A}_\tau^f \times \mathcal{A}_\tau^f \to \mathcal{P}^*(\mathcal{A}_\tau^{st})$ be given by

$x \,\tilde{\|}\, y = x \,\tilde{\lfloor\!\lfloor}\, y + y \,\tilde{\lfloor\!\lfloor}\, x$

$\varepsilon \,\tilde{\lfloor\!\lfloor}\, y = \{y\}$, $\perp \,\tilde{\lfloor\!\lfloor}\, y = \{\perp\}$, $\alpha x \,\tilde{\lfloor\!\lfloor}\, y = \alpha(x \,\tilde{\|}\, y)$.

Define $\tilde{\|}$ on $\mathcal{P}^*(\mathcal{A}_\tau^{st})$ as the extended and lifted version of $\|$ on $\mathcal{A}_\tau^f$.

In order to relate $\mathcal{D}(d \mid s)$ and $\tilde{\mathcal{D}}_d(s)$, to be defined in a minute, we need an auxiliary result stating the distributivity of ρ over the operators. The proof of this result makes use of the structure of $\mathcal{A}_\tau^{st}$ where each element x can be obtained as the least upper bound of the chain of its finite approximations $x[n]$.

(5.4) LEMMA *For $* \in \{\cdot, +, \|\}$ it holds that* $P \circ \tilde{*} = * \circ P$ *on* $\mathcal{P}^*(\mathcal{A}_\tau^{st})$.

PROOF By lemma 2.5 it suffices to show $P \circ \tilde{*} = * \circ \rho$. We only treat the case where $*$ equals $\cdot$ using stream induction:

Let $\tilde{x}, \tilde{y} \in \mathcal{A}_\tau^f$. Suppose $\rho(\tilde{x}) = x$, $\rho(\tilde{y}) = y$. Then we derive

(i) $P(\varepsilon \,\tilde{\cdot}\, \tilde{y}) = P(\{\tilde{y}\}) = \{y\} = \varepsilon \cdot y = \rho(\varepsilon) \cdot \rho(\tilde{y})$;

(ii) $P(\perp \tilde{\cdot} \tilde{y}) = P(\{\perp\}) = \{\perp\} = \perp \cdot y = \rho(\perp) \cdot \rho(\tilde{y})$;

(iii) $P(a\tilde{x} \tilde{\cdot} \tilde{y}) = P(a(\tilde{x} \tilde{\cdot} \tilde{y})) = a P(\tilde{x} \tilde{\cdot} \tilde{y}) = a(\rho(\tilde{x}) \cdot \rho(\tilde{y})) = a\rho(\tilde{x}) \cdot \rho(\tilde{y}) = \rho(a\tilde{x}) \cdot \rho(\tilde{y})$;

(iv) $P(\tau\tilde{x} \tilde{\cdot} \tilde{y}) = P(\tau(\tilde{x} \tilde{\cdot} \tilde{y})) = P(\tilde{x} \tilde{\cdot} \tilde{y}) = \rho(\tilde{x}) \cdot \rho(\tilde{y}) = \rho(\tilde{x}) \cdot \rho(\tilde{y}) = \rho(\tau\tilde{x}) \cdot \rho(\tilde{y})$ using properties of pre-fixing.

By a continuity argument we arrive at $P(\tilde{x} \tilde{\cdot} \tilde{y}) = \rho(\tilde{x}) \cdot \rho(\tilde{y})$ for arbitrary $\tilde{x}, \tilde{y} \in \mathcal{A}^{st}_{\tau}$, since

$P(\tilde{x} \tilde{\cdot} \tilde{y})$

$= P((\mathit{lub}_n\ \tilde{x}[n]) \tilde{\cdot} (\mathit{lub}_n\ \tilde{y}[n]))$

$= \mathit{lub}_n\ P(\tilde{x}[n] \tilde{\cdot} \tilde{y}[n])$ by continuity of P and $\tilde{\cdot}$

$= \mathit{lub}_n\ \rho(\tilde{x}[n]) \cdot \rho(\tilde{y}[n])$ by finiteness of $\tilde{x}[n], \tilde{y}[n]$ for $n \in \mathbb{N}$

$= \rho(\mathit{lub}_n\ \tilde{x}[n]) \cdot \rho(\mathit{lub}_n\ \tilde{y}[n])$ by continuity of ρ and $\cdot$

$= \rho(\tilde{x}) \cdot \rho(\tilde{y})$. □

Next we mimic the denotational mapping $\mathcal{D}$. Note the occurrence of τ in the definition of the transformation $\tilde{H}_d$. This idea of guarding procedure calls with a dummy step can also be observed in the semantics of [BBKM]a.

(5.5) DEFINITION Define the collection $\mathit{Env}\tilde{}$, with typical element $\tilde{\eta}$, by $\mathit{Env}\tilde{} = \mathcal{X} \to \mathcal{P}^*(\mathcal{A}^{st}_{\tau})$. Define the mapping $\tilde{\mathcal{S}}$: $\mathit{Stat}_E \to \mathit{Env}\tilde{} \to \mathcal{P}^*(\mathcal{A}^{st}_{\tau})$ by

$\tilde{\mathcal{S}}(E)(\tilde{\eta}) = \{\varepsilon\}$,

$\tilde{\mathcal{S}}(a)(\tilde{\eta}) = \{a\}$,

$\tilde{\mathcal{S}}(x)(\tilde{\eta}) = \tilde{\eta}(x)$,

$\tilde{\mathcal{S}}(s_1 * s_2)(\tilde{\eta}) = \tilde{\mathcal{S}}(s_1)(\tilde{\eta}) \tilde{*} \tilde{\mathcal{S}}(s_2)(\tilde{\eta})$ for $* \in \{\cdot, +, \|\}$.

Let for $d \in \mathit{Decl}$ the transformation $\tilde{H}_d$: $\mathit{Env}\tilde{} \to \mathit{Env}\tilde{}$ be given by

$\tilde{H}_d(\tilde{\eta})(x) = \tau \cdot \tilde{\mathcal{S}}(s)(\tilde{\eta})$

where $s = d(x)$. For $d \in \mathit{Decl}$ the semantical mapping $\tilde{\mathcal{D}}_d$: $\mathit{Stat}_E \to \mathcal{P}^*(\mathcal{A}^{st}_{\tau})$ is defined by

$\tilde{\mathcal{D}}_d(\bar{s}) = \tilde{\mathcal{S}}(\bar{s})(\tilde{\eta}_d)$

where $\tilde{\eta}_d$ is the least fixed point of $\tilde{H}_d$.

Well-definedness of $\tilde{\mathcal{D}}_d$ is straightforward to check (as for $\mathcal{D}$) and left to the reader. We exploit the Fixed Point Transformation Lemma 2.6 when relating η_d with $\tilde{\eta}_d$ which is crucial to comparing $\mathcal{D}$ with its concrete version $\tilde{\mathcal{D}}$.

(5.6) LEMMA

(i) For $s \in \mathit{Stat}$ it holds that $\rho \circ \tilde{\mathcal{S}}(s) = \mathcal{S}(s) \circ \rho$ with ρ: $\mathcal{P}^*(\mathcal{A}^{st}_{\tau}) \to \mathcal{P}^*(\mathcal{A}^{st})$ and ρ: $\mathit{Env}\tilde{} \to \mathit{Env}$, respectively.

(ii) For $d \in \mathit{Decl}$ it holds that $\tilde{H}_d$ is canonical with retract H_d and $\rho(\tilde{\eta}_d) = \eta_d$.

(iii) For $d \in Decl$ and $s \in Stat$ it holds that $\rho(\tilde{\mathcal{D}}_d(s)) = \mathcal{D}(d\,|\,s)$.

PROOF

(i) By structural induction on s using lemma 5.4.

(ii) Directly from $H_d \circ \rho = \rho \circ \tilde{H}_d$ (where $\rho = \lambda\tilde{\eta}.\lambda x.\rho(\tilde{\eta}(x))$) and the Fixed Point Transformation Lemma.

(iii) Immediate consequence of (i) and (ii). □

The last step to take is proving the equivalence of $\tilde{\mathcal{O}}_d$ and $\tilde{\mathcal{D}}_d$. For this we follow the approach as proposed in [KR]a and [BM]a in the setting of complete metric spaces. First we introduce the higher-order transformation $\tilde{\Phi}_d$ of which both $\tilde{\mathcal{O}}$ and $\tilde{\mathcal{D}}$ will be fixed points.

(5.7) DEFINITION Let $d \in Decl$. Define the higher-order transformation $\tilde{\Phi}_d$: $(Stat_E \to \mathcal{P}^*(\mathcal{A}^{st}_\tau)) \to (Stat_E \to \mathcal{P}^*(\mathcal{A}^{st}_\tau))$ by

$$\tilde{\Phi}_d(M)(E) = \{\varepsilon\},$$

$$\tilde{\Phi}_d(M)(s) = min(\ \cup\ \{\ \alpha \cdot M(\bar{s})\ |\ \ s \to^{\alpha}_d \bar{s}\ \}\ )$$

for meanings $M \in Stat_E \to \mathcal{P}^*(\mathcal{A}^{st}_\tau)$.

By its continuity $\tilde{\Phi}_d$ has a least fixed point, say $\tilde{\phi}_d$. The pivotal property of $\tilde{\Phi}_d$ however is that it has exactly *one* fixed point. This can be checked as follows: Since we are using the Egli-Milner ordering it suffices to show that for all $\bar{s} \in Stat_E$ the image $\tilde{\phi}_d(\bar{s})$ has only maximal elements. For, if this is the case, $\tilde{\phi}_d$ itself is maximal in $Stat_E \to \mathcal{P}^*(\mathcal{A}^{st}_\tau)$ and thus is the unique fixed point of $\tilde{\Phi}_d$.

(5.8) LEMMA *For $d \in Decl$ the transformation $\tilde{\Phi}_d$ has a unique fixed point.*

PROOF Let $\tilde{\phi}_d$ be the least fixed point of $\tilde{\Phi}_d$. It suffices to show $\forall \bar{s} \in Stat_E$: $\tilde{\phi}_d(\bar{s}) \subseteq \mathcal{A}^* \cup \mathcal{A}^\omega$. Suppose this is not the case. Let $n \in \mathbb{N}$ be minimal such that there exist $s \in Stat_E$ and $\alpha_1 \cdot\cdot \alpha_n \perp \in \tilde{\phi}_d(s)$. Note that $s \neq E$ and $n > 0$. By the fixed point property of $\tilde{\phi}_d$ we have $\alpha_1 \cdot\cdot \alpha_n \perp \in \tilde{\Phi}_d(\tilde{\phi}_d)(s) = min\{\ \alpha \cdot \tilde{\phi}_d(\bar{s})\ |\ \ s \to^{\alpha}_d \bar{s}\ \}$. Thus for some $\bar{s} \in Stat_E$ we have $\alpha_2 \cdot\cdot \alpha_n \perp \in \tilde{\phi}_d(\bar{s})$ contradicting the minimality of n. Conclusion $\forall \bar{s} \in Stat_E$: $\tilde{\phi}_d(\bar{s})$ is maximal in $\mathcal{P}^*(\mathcal{A}^{st}_\tau)$. □

It remains to show that both $\tilde{\mathcal{O}}_d$ and $\tilde{\mathcal{D}}_d$ are fixed points of $\tilde{\Phi}_d$, since by the previous lemma the result $\tilde{\mathcal{O}}_d = \tilde{\mathcal{D}}_d$ then follows immediately.

(5.9) THEOREM *For $d \in Decl$ it holds that $\tilde{\mathcal{O}}_d = \tilde{\mathcal{D}}_d$.*

PROOF We check (i) $\tilde{\Phi}_d(\tilde{\mathcal{O}}_d) = \tilde{\mathcal{O}}_d$ and (ii) $\tilde{\Phi}_d(\tilde{\mathcal{D}}_d) = \tilde{\mathcal{D}}_d$.

(i) Clearly $\tilde{\Phi}_d(\tilde{\mathcal{O}}_d)(E) = \tilde{\mathcal{O}}_d(E)$. Let $s \in Stat$. Then $\tilde{\Phi}_d(\tilde{\mathcal{O}}_d)(s) = min(\ \cup\ \{\ \alpha \cdot \tilde{\mathcal{O}}_d(\bar{s})\ |\ \ s \to^{\alpha}_d \bar{s}\ \}\ ) = \tilde{\mathcal{O}}_d(s)$ since $s \in Stat$ always admits a $\to_d$-step.

(ii) By structural induction on s we check $\tilde{\Phi}_d(\tilde{\mathcal{D}}_d) = \tilde{\mathcal{D}}_d$. We only consider the cases x and $s_1 \| s_2$. Suppose $d(x) = s$. Then we have

$\tilde{\Phi}_d(\tilde{\mathcal{D}}_d)(x)$

$= min(\ \cup\ \{\ \alpha \cdot \tilde{\mathcal{D}}_d(\bar{s})\ |\ \ x \rightarrow_d^\alpha \bar{s}\ \}\ )$ (by definition of $\tilde{\Phi}_d$)

$= \tau \cdot \tilde{\mathcal{D}}_d(s)$ (since $x \rightarrow_d^\alpha \bar{s} \Leftrightarrow \alpha = \tau\ \&\ \bar{s} = s$)

$= \tau \cdot \tilde{\mathcal{S}}(s)(\tilde{\eta}_d)$ (by definition of $\tilde{\mathcal{D}}_d$)

$= \tilde{H}_d(\tilde{\eta}_d)(x)$ (since $d(x) = s$)

$= \tilde{\eta}_d(x)$ (since $\tilde{\eta}_d$ is the least fixed point of $\tilde{H}_d$)

$= \tilde{\mathcal{S}}(x)(\tilde{\eta}_d)$ (by definition of $\tilde{\mathcal{S}}$)

$= \tilde{\mathcal{D}}_d(x)$.

We have that

$\tilde{\Phi}_d(\tilde{\mathcal{D}}_d)(s_1 \| s_2)$

$= min(\ \cup\ \{\ \alpha \cdot \tilde{\mathcal{D}}_d(\bar{s})\ |\ \ s_1 \| s_2 \rightarrow_d^\alpha \bar{s}\ \}\ )$ (by definition of $\tilde{\Phi}_d$)

$= min(\ \cup\ \{\ \alpha \cdot \tilde{\mathcal{D}}(\bar{s}_1 \| s_2)\ |\ \ s_1 \rightarrow_d^\alpha \bar{s}_1\ \} \cup \cup\ \{\ \alpha \cdot \tilde{\mathcal{D}}(s_1 \| \bar{s}_2)\ |\ \ s_2 \rightarrow_d^\alpha \bar{s}_2\ \}\ )$
(by inspection of the transition system)

$= min(\ \cup\ \{\ \alpha \cdot \tilde{\mathcal{D}}(\bar{s}_1 \| s_2)\ |\ \ s_1 \rightarrow_d^\alpha \bar{s}_1\ \}\ )\ \tilde{+}\ min(\ \cup\ \{\ \alpha \cdot \tilde{\mathcal{D}}(s_1 \| \bar{s}_2)\ |\ \ s_2 \rightarrow_d^\alpha \bar{s}_2\ \}\ )$
(by definition of $\tilde{+}$ and monotonicity of $\cup$)

$= min(\ \cup\ \{\ \alpha \cdot (\tilde{\mathcal{D}}_d(\bar{s}_1) \tilde{\|} \tilde{\mathcal{D}}_d(s_2))\ |\ \ s_1 \rightarrow_d^\alpha \bar{s}_1\ \}\ )\ \tilde{+}\ min(\ \cup\ \{\ \alpha \cdot (\tilde{\mathcal{D}}_d(s_1) \tilde{\|} \tilde{\mathcal{D}}_d(\bar{s}_2))\ |\ \ s_2 \rightarrow_d^\alpha \bar{s}_2\ \}\ )$ (compositionality of $\tilde{\mathcal{D}}_d$)

$= (\ min(\ \cup\ \{\ \alpha \cdot \tilde{\mathcal{D}}_d(\bar{s}_1)\ |\ \ s_1 \rightarrow_d^\alpha \bar{s}_1\ \}\ )\ \tilde{\mathbb{L}}\ \tilde{\mathcal{D}}_d(s_2))\ \tilde{+}\ (\ min(\ \cup\ \{\ \alpha \cdot \tilde{\mathcal{D}}_d(\bar{s}_2)\ |\ \ s_2 \rightarrow_d^\alpha \bar{s}_2\ \}\ )\ \tilde{\mathbb{L}}\ \tilde{\mathcal{D}}_d(s_1))$ (property of $\tilde{\mathbb{L}}$)

$= \tilde{\Phi}_d(\tilde{\mathcal{D}}_d)(s_1)\ \tilde{\mathbb{L}}\ \tilde{\mathcal{D}}_d(s_2)\ \tilde{+}\ \tilde{\Phi}_d(\tilde{\mathcal{D}}_d)(s_2)\ \tilde{\mathbb{L}}\ \tilde{\mathcal{D}}_d(s_1)$ (by definition of $\tilde{\Phi}_d$)

$= \tilde{\mathcal{D}}_d(s_1)\ \tilde{\mathbb{L}}\ \tilde{\mathcal{D}}_d(s_2)\ \tilde{+}\ \tilde{\mathcal{D}}_d(s_2)\ \tilde{\mathbb{L}}\ \tilde{\mathcal{D}}_d(s_1)$ (by the induction hypothesis)

$= \tilde{\mathcal{D}}_d(s_1) \tilde{\|} \tilde{\mathcal{D}}_d(s_2)$ (by definition of $\tilde{\|}$)

$= \tilde{\mathcal{D}}_d(s_1 \| s_2)$ (by definition of $\tilde{\mathcal{D}}_d$). □

Taking all together, viz. lemma 5.2, lemma 5.6 and theorem 5.9, we have shown $min(\ \mathcal{O}(d \mid s)\ ) = \mathcal{D}(d \mid s)$.

References

[BV] A. de Bruin and E.P. de Vink, ‘‘Retractions in Comparing Prolog Semantics,’’ pp. 71-90 in *Proc. Computing Science in the Netherlands, Part 1*, P.M.G. Apers, D. Bosman & J. van Leeuwen (eds.), Utrecht (1989).

[BKMOZ] J.W. de Bakker, J.N. Kok, J.-J.Ch. Meyer, E.-R. Olderog, and J.I. Zucker, ‘‘Contrasting Themes in the Semantics of Imperative Concurrency,’’ pp. 51-121 in *Current Trends in Concurrency: Overviews and Tutorials*, J.W. de Bakker, W.P de Roever & G. Rozenberg (eds.), LNCS **224**, Springer (1986).

[Me] J.-J.Ch. Meyer, *Programming Calculi Based on Fixed Point Transformations: Semantics and Applications*, Dissertation, Vrije Universiteit, Amsterdam (1985).

[Pl] G.D. Plotkin, ‘‘A Powerdomain Construction,’’ *SIAM Journal of Computing* **5**, pp. 452-

487 (1976).

[MV] J.-J.Ch. Meyer and E.P. de Vink, ‘‘Applications of Compactness in the Smyth Powerdomain of Streams,’’ *Theoretical Computer Science* **57**, pp. 251-282 (1988).

[Sm] M.B. Smyth, ‘‘Powerdomains,’’ *Journal of Computer System Sciences* **16**, pp. 23-26 (1978).

[BHR] S.D. Brookes, C.A.R. Hoare, and A.W. Roscoe, ‘‘A Theory of Communicating Sequential Processes,’’ *Journal of the ACM* **31**, pp. 560-599 (1984).

[KR] J.N. Kok and J.J.M.M. Rutten, ‘‘Contractions in Comparing Concurrency Semantics,’’ pp. 317-332 in *Proc. ICALP’88*, T. Lepistö & A. Salomaa (eds.), LNCS **317**, Springer (1988).

[BBKM] 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**, pp. 135-156 (1984).

[Vi2] E.P. de Vink, *Designing Stream Based Semantics for Uniform Concurrency and Logic Programming*, Dissertation, Vrije Universiteit, Amsterdam (1990).

[BMZ] J.W. de Bakker, J.-J.Ch Meyer, and J.I Zucker, ‘‘On Infinite Computations in Denotational Semantics,’’ *Theoretical Computer Science* **26**, pp. 53-82 (1983).

[HP] M. Hennessy and G.D. Plotkin, ‘‘Full Abstraction for a Simple Parallel Programming Language,’’ pp. 108-120 in *Proc. 8th MFCS*, J. Bečvař (ed.), LNCS **74**, Springer (1979).

[Ap] K.R. Apt, ‘‘Recursive Assertions and Parallel Programs,’’ *Acta Informatica* **15**, pp. 219-232 (1983).

[Mi] R. Milner, *A Calculus of Communicating Systems*, LNCS **92**, Springer (1980).

[BK] J.A. Bergstra and J.W. Klop, ‘‘Algebra of Communicating Processes,’’ pp. 89-138 in *Proc. CWI Symposium on Mathematics and Computer Science*, J.W. de Bakker, M. Hazewinkel & J.K. Lenstra (eds.), CWI Monograph I (1986).

[BMOZ] J.W. de Bakker, J.-J.Ch. Meyer, E.-R. Olderog, and J.I. Zucker, ‘‘Transition Systems, Metric Spaces and Ready Sets in the Semantics of Uniform Concurrency,’’ *Journal of Computer and System Sciences*, pp. 158-224 (1988).

[BM] J.W. de Bakker and J.-J.Ch. Meyer, ‘‘Metric Semantics for Concurrency,’’ *BIT* **28**, pp. 504-529 (1988).

[Vi1] E.P. de Vink, ‘‘Comparative Semantics for Prolog with Cut,’’ *Science of Computer Programming* **13**, pp. 237-264 (1989/90).

[Ba] J.W. de Bakker, *Mathematical Theory of Program Correctness*, Prentice Hall International, London (1980).