Retractions in Comparing Prolog Semantics

A. de Bruin

Faculty of Economics,
Erasmus Universiteit, P.O.Box 1738, NL-3000 DR Rotterdam
E.P. de Vink

Department of Mathematics and Computer Science,
Vrije Universiteit, De Boelelaan 1081, NL-1081 HV Amsterdam

ABSTRACT We present an operational model $\mathcal{O}$ and a continuation based denotational model $\mathcal{D}$ for a uniform variant of Prolog, including the cut operator. The two semantical definitions make use of higher order transformations Φ and Ψ, respectively. We prove $\mathcal{O}$ and $\mathcal{D}$ equivalent in a novel way by comparing yet another pair of higher order transformations $\tilde{\Phi}$ and $\tilde{\Psi}$, that yield Φ and Ψ, respectively, by application of a suitable abstraction operator.

Section 1 Introduction

In [BV] we presented both an operational and a denotational continuation based semantics for the core of Prolog, and we proved these two semantics equivalent. We used a two step approach, by first deriving these results for an intermediate language, obtained by stripping the logic programming aspects (substitutions, most general unifiers and all that) from Prolog. This resulted in the abstract language $\mathcal{B}$ in which only the control structure from Prolog remained, such as the backtrack mechanism and the cut operator. After having compared the operational and denotational meanings for $\mathcal{B}$ successfully we generalized as a next step the two semantics to the case of Prolog while preserving their equivalence.

The language $\mathcal{B}$ will be investigated again in this paper, but now more as a guinea pig. We will use it to test a new idea for proving equivalence of operational and denotational semantics based on cpo's. The main virtue of $\mathcal{B}$ in this respect is that although it is a sequential language it has a nontrivial control structure. In fact, the denotational semantics of this language needs three continuations to adequately describe the flow of control.

We will discuss our new approach to equivalence proofs by comparing it with the standard way these proofs have been conducted so far. To this end, we first spend a few words on operational semantics. The main idea behind this brand of semantics is to describe how an abstract machine executes a program in the language of interest. The abstract machine is defined by specifying the configurations it can be in, and by introducing a step function, mapping configurations to configurations, thus describing the behavior of the abstract machine. Starting from an initial configuration C_o, repeatedly applying the step function will deliver a number of intermediate configurations C_o, C_1, C_2, with $C_{i+1} = step\ C_i$. The computation

terminates when a final configuration has been reached.

Our operational meaning function $\mathcal{O}$ abstracts from these intermediate results however, defining the meaning $\mathcal{O}[\![s]\!]$ of a statement s as a state transformation, mapping initial states to final states while recording the wanted observations. The final state is obtained by iterating the step function and this iteration can nicely be captured by taking a fixed point of a suitable higher order operator Φ.

Notice that, due to the abstraction of (most of) the intermediate configurations we made, in general the operator Φ will not have a unique fixed point. For instance, suppose iterating the step function fails to produce a final configuration, i.e., we deal with a nonterminating computation. In that case we put $\mathcal{O}[\![s]\!] = \bot$. However, other fixed points of Φ are possible, yielding different results for such s. The fact that fixed points are not unique complicates matters when it comes to prove $\mathcal{O}$ equivalent to $\mathcal{D}$, the denotational meaning function. The standard technique in such a proof is to show that a step of the abstract machine does not affect the denotational meaning of the configurations being transformed. More technically, if we have that *step* $C = C'$, and if we extend $\mathcal{D}$ to a function $\mathcal{I}$ taking configurations as arguments, then we have to show hat $\mathcal{I}C = \mathcal{I}C'$. From this result we will then be able to infer hat $\mathcal{I}C_{initial} = \mathcal{I}C_{final}$, by induction on the number of steps needed. We then deduce that ($\$$) $\mathcal{D}[\![s]\!]\sigma = \mathcal{O}[\![s]\!]\sigma$.

However, there is a flaw in this line of reasoning. The result ($\$$) is only valid for terminating computations. If iterating the step function does not produce a final configuration then the above argument does not work. This means that in order to complete the equivalence proof $\mathcal{D} = \mathcal{O}$, we have to derive the result that $\mathcal{O}[\![s]\!]\sigma = \bot$ implies $\mathcal{D}[\![s]\!]\sigma = \bot$. Unfortunately this takes at least as much effort as was needed to derive the previous result. See for example the proofs in [Ba1].

On the other hand if the operator Φ would have a unique fixed point then it would not be necessary to derive this additional result. Uniqueness is guaranteed for instance when one does not use cpo's and a continuity argument to ensure the existence of the fixed points, but when complete metric spaces are used instead and the operators are contracting functions on these spaces. For in that case Banach's theorem can be applied. In [KR] unicity of the fixed point of the operational higher order operator Φ has been exploited successfully to derive compact equivalence proofs for operational and denotational meanings along the lines sketched above. A similar line of reasoning has been used in [BM]. The fact that our operator Φ admits more than one fixed point seems to be an essential consequence due to the fact that we abstracted away from the intermediate configurations. For instance, it is not possible to define in a straightforward way a contraction on metric spaces which yields the same operational semantics.

The idea behind our equivalence proof is to introduce a slightly less abstract operational semantics using a new operator $\tilde{\Phi}$ that does have a unique fixed point. Our semantics is made more concrete because it does not simply deliver observables σ as the result of a

computation, but also additional information. The resulting states σ are preceded by a number of clock ticks, a row of τ's. (In algebraic approaches as e.g. [Mi] and [BeKl], τ denotes a silent or internal step of a process.) Here the idea is that each τ in this row corresponds with the execution of an elementary action by the abstract machine, i.e. one iteration of the step function. Similarly for a nonterminating computation, we do not deliver $\perp$ but an infinite row of τ's instead. So $\perp$ is reformulated as internal divergence, (as in [BBKM]). Now, for the corresponding meaning functions $\tilde{\mathcal{O}}$ and $\tilde{\mathcal{D}}$ a compact equivalence proof can be given. In order to establish from this the equivalence of our original functions $\mathcal{O}$ and $\mathcal{D}$ it is sufficient to show that there exists an abstraction operator *strip*, "a τ-remover" so to speak, such that for all s and σ we have (*) $\mathcal{O}[\![s]\!]\sigma = strip(\tilde{\mathcal{O}}[\![s]\!]\sigma)$ and $\mathcal{D}[\![s]\!]\sigma = strip(\tilde{\mathcal{D}}[\![s]\!]\sigma)$.

In the next section we will show that this idea can be made to work for a simple language, the most complicated construct of which is the while statement. However it will also become clear that the complexity of the over all proof has not diminished. This is caused by the fact that substantial additional work has to be performed in proving the equalities (*) above. On the other hand, the reasoning in these proofs is to a great extent independent of the particular language investigated. The result $\mathcal{O}[\![s]\!]\sigma = strip(\tilde{\mathcal{O}}[\![s]\!]\sigma)$ is valid for each operational semantics derived from (deterministic) step functions. In the proof of $\mathcal{D}[\![s]\!]\sigma = strip(\tilde{\mathcal{D}}[\![s]\!]\sigma)$ a number a generic elements seems to be present as well, which can be carried over unaltered to similar proofs for other languages. This observation is worked out in section 3, where we present a more general theory on the relation between abstract domains and concretizations thereof like the ones discussed above. We show that the abstract domains can be considered as so called retracts of the more concrete ones. We will derive a few theorems, related to those of [BMZ] and [Me] that enable us to prove results as in (*) in a more smooth and elegant way. In the remaining sections this theory is tested using the above described language $\mathcal{B}$: Section 4 describes the operational and denotational semantics of $\mathcal{B}$. Section 5 will be devoted to the actual equivalence proof.

Acknowledgements. From the above it will be clear that our work relies heavily on that of others. The immediate starting point of this paper is [KR]a where for the first time the observation was made that compact equivalence proofs could be realized using higher order transformations. To a great extent, we also benefit from the work on metric semantics of concurrency performed by De Bakker e.a. E.g. [BZ], [BKMOZ], [BM]a, [Ba2]. It is a pleasure to thank the forum formed by the members of the working group on concurrency, - Jaco de Bakker, Frank de Boer, Joost Kok, Jan Rutten and others - for their comments and the good scientific atmosphere they provided. We are also indebted to Philippe Darondeau for his suggestions for improvement of the manuscript. Finally we are grateful to M279 for her hospitality.

Section 2 A Simple Example: the While Statement

In this section we will illustrate the basic idea behind our new equivalence proof, by sketching how such a proof can be given for the very simple language defined below.

(2.1) DEFINITION The set of elementary statements *EStat* is defined by $e ::= x{:=}t \mid \mathit{if}\ b\ \mathit{then}\ s_1\ \mathit{else}\ s_2\ \mathit{fi} \mid \mathit{while}\ b\ \mathit{do}\ s\ \mathit{od}$. The set of statements *Stat* is defined by $s ::= \varepsilon \mid e;s$.

As remarked in section 1, an operational semantics is defined via an abstract machine. Such a machine, called a transition system, can be specified by giving its configurations together with a relation between configurations which describes the step function. To avoid too many details we assume the existence of an interpretation I from which the effect of executing an assignment statement can be obtained, as well as the value of boolean expressions. Notice that there is no transition defined from configurations of the form $[\ \varepsilon, \sigma\ ]$. These are the final configurations corresponding to terminated computations.

(2.2) DEFINITION

(i) The set of configurations *Conf* is defined as the collection of statement-state pairs $[\ s,\sigma\ ]$, i.e. $\mathit{Conf} = \{\ [\ s,\sigma\ ] \mid s \in \mathit{Stat}, \sigma \in \Sigma\ \}$.

(ii) The step function $\rightarrow$ is the smallest subset of $\mathit{Conf} \times \mathit{Conf}$ such that

$[\ x{:=}t;s,\sigma\ ] \rightarrow [\ s,\sigma'\ ]$ where $\sigma' = I[\![\ x{:=}t\ ]\!]\ \sigma$

$[\ \mathit{if}\ b\ \mathit{then}\ s_1\ \mathit{else}\ s_2\ \mathit{fi};s,\sigma\ ] \rightarrow [\ s_1;s,\sigma\ ]$ if $I[\![\ b\ ]\!]\ \sigma = \mathit{tt}$

$[\ \mathit{if}\ b\ \mathit{then}\ s_1\ \mathit{else}\ s_2\ \mathit{fi};s,\sigma\ ] \rightarrow [\ s_2;s,\sigma\ ]$ if $I[\![\ b\ ]\!]\ \sigma = \mathit{ff}$

$[\ \mathit{while}\ b\ \mathit{do}\ s'\ \mathit{od};s,\sigma\ ] \rightarrow [\ s';\mathit{while}\ b\ \mathit{do}\ s'\ \mathit{od};s,\sigma\ ]$ if $I[\![\ b\ ]\!]\ \sigma = \mathit{tt}$

$[\ \mathit{while}\ b\ \mathit{do}\ s'\ \mathit{od};s,\sigma\ ] \rightarrow [\ s,\sigma\ ]$ if $I[\![\ b\ ]\!]\ \sigma = \mathit{ff}$

(iii) The operational semantics $\mathcal{O} : \mathit{Stat} \rightarrow \Sigma \rightarrow \Sigma_\perp$ is defined by $\mathcal{O}[\![\ s\ ]\!]\ \sigma = \mu\Phi([\ s,\sigma\ ])$ where $\Phi : [\ (\mathit{Conf} \rightarrow \Sigma_\perp) \rightarrow (\mathit{Conf} \rightarrow \Sigma_\perp)\ ]$ is given by $\Phi O[\ \varepsilon, \sigma\ ] = \sigma$, $\Phi OC = OC'$ if $C \rightarrow C'$, and $\Sigma_\perp$ denotes the flat cpo with least element $\perp$.

Our denotational semantics will use continuations. Although this seems to be a bit too heavy for such a simple language, we do not use direct semantics for two reasons. First of all, the equivalence proof will proceed more smoothly when using continuations and secondly, the language $\mathcal{B}$ - that will be studied in the sequel - cannot be given a satisfactory direct semantics. (In order to model the cut operator this way one has to resort to cut-flags or other kinds of indicators. See [JM], [DM], [Bd], [Vi].)

Our meaning function $\mathcal{D}$ is defined as the least fixed point of a higher order operator Ψ. In fact, it does not matter much how $\mathcal{D}$ is defined (as long as it remains denotational), the more usual approach based on environments as in [Ba1]a would work equally well, cf.

[BM]a. The style of defining here is closer to [KR]a which we take as a starting point.

(2.3) DEFINITION

(i) The collections *Cont* of continuations and the set *Meaning* of meanings are given by $Cont = \Sigma \to \Sigma_\perp$ and $Meaning = Stat \to [\, Cont \to \Sigma \to \Sigma_\perp \,]$.

(ii) The denotational semantics $\mathcal{D} : Stat \to \Sigma \to \Sigma^{st}$ is defined by $\mathcal{D}[\![s]\!] \sigma = \mu\Psi[\![s]\!] \xi_o \sigma$ where $\xi_o = \lambda\sigma.\sigma$ and $\Psi : [\, Meaning \to Meaning \,]$ is given by

$\Psi M[\![\varepsilon]\!] \xi\sigma = \xi\sigma,$

$\Psi M[\![x{:=}t]\!] \xi\sigma = \xi\sigma'$ where $\sigma' = I[\![x{:=}t]\!] \sigma,$

$\Psi M[\![\mathit{if}\ b\ \mathit{then}\ s_1\ \mathit{else}\ s_2\ \mathit{fi}]\!] \xi\sigma =$

$M[\![s_1]\!] \xi\sigma$ if $I[\![b]\!] \sigma = tt,$

$\Psi M[\![\mathit{if}\ b\ \mathit{then}\ s_1\ \mathit{else}\ s_2\ \mathit{fi}]\!] \xi\sigma =$

$M[\![s_2]\!] \xi\sigma$ if $I[\![b]\!] \sigma = ff,$

$\Psi M[\![\mathit{while}\ b\ \mathit{do}\ s\ \mathit{od}]\!] \xi\sigma =$

$M[\![s]\!] \{ M[\![\mathit{while}\ b\ \mathit{do}\ s\ \mathit{od}]\!] \xi \} \sigma$ if $I[\![b]\!] \sigma = tt,$

$\Psi M[\![\mathit{while}\ b\ \mathit{do}\ s\ \mathit{od}]\!] \xi\sigma =$

$\xi\sigma$ if $I[\![b]\!] \sigma = ff,$

$\Psi M[\![e;s]\!] \xi\sigma = M[\![e]\!] \{ M[\![s]\!] \xi \} \sigma.$

Before giving the new equivalence proof, we discuss the old approach for a while. First of all one proves $\mathcal{O} \le \mathcal{D}$. The idea is to extend $\mathcal{D}$ to a intermediate function $\mathcal{I}$ defined on configurations: $\mathcal{I}[\, s, \sigma \,] = \mathcal{D}[\![s]\!] \sigma$. One then proves $\Phi\mathcal{I} = \mathcal{I}$, in essence by checking this for all possible configurations (cf. 2.2.ii, and the proof of 2.7). From this result $\mathcal{O} \le \mathcal{D}$ follows immediately, but in fact we have more than that. Because $\mu\Phi$ delivers results in $\Sigma_\perp$, a flat domain, we have that for all s and σ such that $\mu\Phi[\, s, \sigma \,] = \sigma' \neq \perp$, also $\mathcal{I}[\, s, \sigma \,] = \sigma'$ holds. Therefore, in order to complete our proof we only need to show $\mathcal{O}[\![s]\!] \sigma = \perp \Rightarrow \mathcal{D}[\![s]\!] \sigma = \perp$. This is most easily accomplished by showing that for all approximations $\Psi^i\perp$ of $\mathcal{D}$ we have $\Psi^i\perp \le \mathcal{O}$, and this can be proved by induction on i, checking all possible forms a statement s can take.

This second half of the equivalence proof would not be needed if Φ would have a unique fixed point: for $\Phi\mathcal{I} = \mathcal{I}$ would then imply $\mathcal{I} = \mu\Phi$ and thus $\mathcal{O} = \mathcal{D}$. The idea now is to make Φ a little bit more concrete, making it deliver its result in the cpo $\tilde{\Sigma}^{st}$ of streams instead of the flat domain $\Sigma_\perp$. This will be accomplished by prefixing a result σ' with a number of τ's, each of which denotes a "clock tick", corresponding with an elementary step of our abstract machine. The effect of all this will be that our new operator $\tilde{\Phi}$ will indeed have a unique fixed point (cf. lemma 2.5).

(2.4) DEFINITION

(i) Put $\tilde{\Sigma} = \Sigma \cup \{\tau\}$ for some distinguished $\tau \notin \Sigma$. $\tilde{\Sigma}$ is ranged over by θ. Let $\tilde{\Sigma}^{st}$ denote the cpo of streams over $\tilde{\Sigma}$, i.e. $\tilde{\Sigma}^{st} = \tilde{\Sigma}^* \cup \tilde{\Sigma}^*.\bot \cup \tilde{\Sigma}^\omega$, and $x <_{st} y \Leftrightarrow \exists x' \in \tilde{\Sigma}^* \; \exists y' \in \tilde{\Sigma}^{st} \setminus \{\bot\}: x = x'.\bot \;\&\; y = x'.y'$. (Cf. [Me]a, [MV].)

(ii) The step function $\rightarrow$ is the smallest subset of $Conf \times \tilde{\Sigma} \times Conf$ such that
$[\, x{:=}t;s,\sigma\,] \rightarrow_\tau [\, s,\sigma'\,]$ where $\sigma' = I[\![\, x{:=}t \,]\!]\,\sigma$
$[\, \mathit{if}\ b\ \mathit{then}\ s_1\ \mathit{else}\ s_2\ \mathit{fi};s,\sigma\,] \rightarrow_\tau [\, s_1;s,\sigma\,]$ if $I[\![\, b \,]\!]\,\sigma = tt$
$[\, \mathit{if}\ b\ \mathit{then}\ s_1\ \mathit{else}\ s_2\ \mathit{fi};s,\sigma\,] \rightarrow_\tau [\, s_2;s,\sigma\,]$ if $I[\![\, b \,]\!]\,\sigma = ff$
$[\, \mathit{while}\ b\ \mathit{do}\ s'\ \mathit{od};s,\sigma\,] \rightarrow_\tau [\, s';\mathit{while}\ b\ \mathit{do}\ s'\ \mathit{od};s,\sigma\,]$ if $I[\![\, b \,]\!]\,\sigma = tt$
$[\, \mathit{while}\ b\ \mathit{do}\ s'\ \mathit{od};s,\sigma\,] \rightarrow_\tau [\, s,\sigma\,]$ if $I[\![\, b \,]\!]\,\sigma = ff$

(iii) The operational semantics $\tilde{\mathcal{O}} : Stat \rightarrow \Sigma \rightarrow \tilde{\Sigma}^{st}$ is defined by $\tilde{\mathcal{O}}[\![\, s \,]\!]\,\sigma = \mu\tilde{\Phi}([\, s,\sigma\,])$ where $\tilde{\Phi} \in [\, (Conf \rightarrow \tilde{\Sigma}^{st}) \rightarrow (Conf \rightarrow \tilde{\Sigma}^{st})\,]$ is given by $\tilde{\Phi}O[\, \varepsilon,\sigma\,] = \sigma$, $\tilde{\Phi}OC = \tau \bullet OC'$ if $C \rightarrow_\tau C'$.

Notice that definitions 2.2(ii) and 2.4(ii) are exactly the same except for the labels τ.

It is instructive to observe the relation between the functions $\mathcal{O}$ and $\tilde{\mathcal{O}}$. We observe, without proof that $\mathcal{O}[\![\, s \,]\!]\,\sigma = \sigma' \Leftrightarrow \exists k \in \mathbb{N}: \tilde{\mathcal{O}}[\![\, s \,]\!]\,\sigma = \tau^k.\sigma'$ and $\mathcal{O}[\![\, s \,]\!]\,\sigma = \bot \Leftrightarrow \tilde{\mathcal{O}}[\![\, s \,]\!]\,\sigma = \tau^\omega$. Of course a similar result is true for $\mu\Phi$ and $\mu\tilde{\Phi}$. Notice that this implies that for all configurations C we have that $\mu\tilde{\Phi}C$ is maximal in $\tilde{\Sigma}^{st}$ and this again means that $\mu\tilde{\Phi}$ itself is maximal. Therefore $\mu\tilde{\Phi}$ is not only the least fixed point of $\tilde{\Phi}$; it is the only one!

(2.5) LEMMA $\tilde{\Phi}$ *has a unique fixed point.*

PROOF We prove that $\mu\tilde{\Phi}$ is maximal, from which it follows that $\mu\tilde{\Phi}$ is unique. The proof is by contradiction. Suppose $\mu\tilde{\Phi}$ is not maximal. Then there exists at least one C with the property that $\mu\tilde{\Phi}C$ is not maximal in $\tilde{\Sigma}^{st}$, i.e. $\mu\tilde{\Phi}C$ must be of the form $x.\bot$. Now choose from the set of all configurations with this property one configuration, say $\bar{C}$, such that $\mu\tilde{\Phi}\bar{C}$ has minimal length.
Because $\mu\tilde{\Phi}$ is a fixed point, we have $\mu\tilde{\Phi}C = \tilde{\Phi}(\mu\tilde{\Phi}C) = \tau.\mu\tilde{\Phi}C'$ for some $C' \in Conf$ such that $C \rightarrow C'$. This however means that $\mu\tilde{\Phi}C'$ is also of the form $x.\bot$, which contradicts the minimality of $|\mu\tilde{\Phi}\bar{C}|$. □

We now have to define a denotational semantics $\tilde{\mathcal{D}}$ which should be equivalent with $\tilde{\mathcal{O}}$. This is done below. Notice that some care has to be taken in adding τ's in the defining clauses of $\tilde{\Psi}$ as compared to the definition of Ψ in 2.3. (Notice furthermore that, although our notation does not show this, the standard continuation ξ_o now delivers a one element stream from $\tilde{\Sigma}^{st}$, whereas in definition 2.3 a single element in $\Sigma_\bot$ was delivered. We tacitly consider $\Sigma_\bot$ as a subcpo of $\tilde{\Sigma}^{st}$.)

(2.6) DEFINITION

(i) The collections $Cont^{\sim}$ of continuations and the set $Meaning^{\sim}$ of meanings are given by $Cont^{\sim} = \Sigma \to \tilde{\Sigma}^{st}$ and $Meaning^{\sim} = Stat \to [\, Cont^{\sim} \to \Sigma \to \tilde{\Sigma}^{st}\,]$.

(ii) The denotational semantics $\tilde{\mathcal{D}} : Stat \to \Sigma \to \tilde{\Sigma}^{st}$ is defined by $\tilde{\mathcal{D}}[\![s]\!]\sigma = \mu\tilde{\Psi}[\![s]\!]\xi_o\sigma$ where $\xi_o = \lambda\sigma.\sigma$ and $\tilde{\Psi} : [\, Meaning^{\sim} \to Meaning^{\sim}\,]$ is given by

$$\tilde{\Psi}M[\![\varepsilon]\!]\xi\sigma = \xi\sigma,$$

$$\tilde{\Psi}M[\![x{:=}t]\!]\xi\sigma = \tau \bullet \xi\sigma' \quad \text{where } \sigma' = I[\![x{:=}t]\!]\sigma,$$

$$\tilde{\Psi}M[\![\textit{if } b \textit{ then } s_1 \textit{ else } s_2 \textit{ fi}]\!]\xi\sigma = \quad \tau \bullet M[\![s_1]\!]\xi\sigma \quad \text{if } I[\![b]\!]\sigma = tt,$$

$$\tilde{\Psi}M[\![\textit{if } b \textit{ then } s_1 \textit{ else } s_2 \textit{ fi}]\!]\xi\sigma = \quad \tau \bullet M[\![s_2]\!]\xi\sigma \quad \text{if } I[\![b]\!]\sigma = ff,$$

$$\tilde{\Psi}M[\![\textit{while } b \textit{ do } s \textit{ od}]\!]\xi\sigma =$$

$$\tau \bullet M[\![s]\!]\{ M[\![\textit{while } b \textit{ do } s \textit{ od}]\!]\xi \}\sigma \quad \text{if } I[\![b]\!]\sigma = tt,$$

$$\tilde{\Psi}M[\![\textit{while } b \textit{ do } s \textit{ od}]\!]\xi\sigma = \quad \tau \bullet \xi\sigma \quad \text{if } I[\![b]\!]\sigma = ff,$$

$$\tilde{\Psi}M[\![e;s]\!]\xi\sigma = M[\![e]\!]\{ M[\![s]\!]\xi \}\sigma.$$

After extending $\tilde{\mathcal{D}}$ into an intermediate function $\tilde{\mathcal{I}}$ acting on configurations we have the following main lemma.

(2.7) LEMMA *Define* $\tilde{\mathcal{I}} : Conf \to \tilde{\Sigma}^{st}$ *by* $\tilde{\mathcal{I}}([\, s,\sigma\,]) = \tilde{\mathcal{D}}[\![s]\!]\sigma$. *Then it holds that* $\tilde{\Phi}\tilde{\mathcal{I}} = \tilde{\mathcal{I}}$.

PROOF We have to prove $\tilde{\Phi}\tilde{\mathcal{I}}C = \tilde{\mathcal{I}}C$ for all configurations C. We only consider the case $C = [\, \textit{while } b \textit{ do } s'\textit{od};s,\sigma\,]$ with $I[\![b]\!]\sigma = tt$. Then on the one hand $\tilde{\Phi}\tilde{\mathcal{I}}C = \tau \bullet \tilde{\mathcal{I}}[\, s';\textit{while } b \textit{ do } s' \textit{ od};s,\sigma\,] = \tau \bullet \tilde{\mathcal{D}}[\![s';\textit{while } b \textit{ do } s' \textit{ od};s]\!]\sigma$, and on the other $\tilde{\mathcal{I}}C = \tilde{\mathcal{D}}[\![\textit{while } b \textit{ do } s' \textit{ od};s]\!]\sigma = \mu\tilde{\Psi}[\![\textit{while } b \textit{ do } s' \textit{ od};s]\!]\xi_o\sigma =$

$$\mu\tilde{\Psi}[\![\textit{while } b \textit{ do } s' \textit{ od}]\!]\{ \mu\tilde{\Psi}[\![s]\!]\xi_o \}\sigma =$$

$$\tau \bullet \mu\tilde{\Psi}[\![s']\!]\{ \mu\tilde{\Psi}[\![\textit{while } b \textit{ do } s' \textit{ od}]\!]\{ \mu\tilde{\Psi}[\![s]\!]\xi_o \} \}\sigma =$$

$$\tau \bullet \mu\tilde{\Psi}[\![s']\!]\{ \mu\tilde{\Psi}[\![\textit{while } b \textit{ do } s' \textit{ od};s]\!]\xi_o \}\sigma = \tau \bullet \mu\tilde{\Psi}[\![s';\textit{while } b \textit{ do } s' \textit{ od};s]\!]\xi_o\sigma =$$

$$\tau \bullet \tilde{\mathcal{D}}[\![s';\textit{while } b \textit{ do } s' \textit{ od};s]\!]\sigma. \quad \square$$

This lemma and lemma 2.5 establish the first part of our proof, viz. $\tilde{\mathcal{O}} = \tilde{\mathcal{D}}$ since this follows from $\mu\tilde{\Phi} = \mathcal{I}$. In order to derive $\mathcal{O} = \mathcal{D}$ we introduce an operator which removes the τ's from the results of $\tilde{\mathcal{O}}$ and $\tilde{\mathcal{D}}$, cf. the remarks following definition 2.4. This abstraction operator *strip* is defined as follows.

(2.8) DEFINITION The function $strip \in [\, \tilde{\Sigma}^{st} \to \Sigma_\perp\,]$ is defined by $strip = \mu P$ where $P \in [\, [\, \tilde{\Sigma}^{st} \to \Sigma_\perp\,] \to [\, \tilde{\Sigma}^{st} \to \Sigma_\perp\,]\,]$ is defined by $P\rho\perp = \perp$, $P\rho\varepsilon = \varepsilon$, $P\rho(\tau.x) = \rho x$, $P\rho(\sigma.x) = \sigma$.

So *strip* yields the first proper state of a stream over $\tilde{\Sigma}$. Notice that this operator P indeed has the functionality as claimed, and that therefore *strip* is continuous. The next two lemma's now furnish the last results needed to prove $\mathcal{O} = \mathcal{D}$.

(2.9) LEMMA *For all s and* σ*:* $\mathcal{O}[\![s]\!]\sigma = strip(\tilde{\mathcal{O}}[\![s]\!]\sigma)$.
PROOF The same result holds for all approximations of $\mathcal{O}$ and $\tilde{\mathcal{O}}$, i.e. we have $\Phi^i\bot = strip(\tilde{\Phi}^i\bot)$. This can be proved by induction on *i*. The lemma now follows from the continuity of *strip*. □

(2.10) LEMMA *For all s and* σ*:* $\mathcal{D}[\![s]\!]\sigma = strip(\tilde{\mathcal{D}}[\![s]\!]\sigma)$.
PROOF We first prove a somewhat stronger fact about the approximations: If for some $\xi \in Cont$, $\tilde{\xi} \in \tilde{Cont}$ we have $\xi = strip \circ \tilde{\xi}$, then for all *i*, *s* and σ: $\Psi^i\bot[\![s]\!]\xi\sigma = strip(\tilde{\Psi}^i\bot[\![s]\!]\tilde{\xi}\sigma)$. This fact can be proved by induction on *i*, checking all possibilities for *s*. As an example we consider the statement *while b do s od*, evaluated in a state in which *b* is true. $\Psi^i\bot[\![$ *while b do s od* $]\!]\xi\sigma = \Psi^{i-1}\bot[\![s]\!]\{\Psi^{i-1}\bot[\![$ *while b do s od* $]\!]\xi\}\sigma$. Now from the induction hypothesis we learn that $\Psi^{i-1}\bot[\![$ *while b do s od* $]\!]\xi = strip(\tilde{\Psi}^{i-1}\bot[\![$ *while b do s od* $]\!]\tilde{\xi})$ and applying the induction hypothesis again, using this fact we infer $\Psi^{i-1}\bot[\![s]\!]\{\Psi^{i-1}\bot[\![$ *while b do s od* $]\!]\xi\} = strip(\tau \bullet \tilde{\Psi}^i\bot[\![$ *while b do s od* $]\!]\tilde{\xi}\sigma)$. □

From the above lemmas and lemma 2.7 we derive the equivalence of the operational and denotational semantics for $\mathcal{B}$ as outlined before.

(2.11) THEOREM $\mathcal{O} = \mathcal{D}$. □

Let us compare the new equivalence proof with the older one discussed after definition 2.3. Indeed, the core of our proof (lemma 2.7) is more compact now, but we had to pay a price: lemmas 2.9 and 2.10 had to be proven as well. For lemma 2.9 this is no big problem though. This proof does not depend on the underlying language, but only on the way the operator Φ has been derived from the transition system. Therefore this lemma can be used for other languages as well, (cf. section 5), it needs to be proven only once.

The proof of lemma 2.10 seems to depend more on the underlying language. At first sight it looks like the work we disposed of in lemma 2.7 now bounces back at us. However in this proof as well there is a language independent part. In order to see this it is worthwhile to study the relation between definition 2.6 and 2.3. In the latter one we inserted τ's while in the former one we do not have such clock ticks. Notice that if we omit the τ's from definition 2.6 we get definition 2.3 back. Notice also that there is a similar relation between definitions 2.4 and 2.2. With some abuse of notation this relation can be written as (2.2) = *strip*(2.4) and (2.3) = *strip*(2.6).

Now these definitions established operators Φ, Ψ, $\tilde{\Phi}$ and $\tilde{\Psi}$, and by taking least fixed points we arrive at the functions $\mathcal{O}$, $\mathcal{D}$, $\tilde{\mathcal{O}}$ and $\tilde{\mathcal{D}}$. For these resultant functions we have proven a similar result as claimed for the definitions: $\mathcal{O} = strip(\tilde{\mathcal{O}})$ and $\mathcal{D} = strip(\tilde{\mathcal{D}})$, (again with some abuse of notation). We would like to have a generic theorem that would provide us with the above relations in one blow: Let $\tilde{\Delta}$ be a definition of some higher order operator $\tilde{\Theta}$ and let Δ be the same definition, only without τ's, defining an operator Θ, (i.e. $\Delta = strip(\tilde{\Delta})$. Then, under certain restrictions on the form of $\tilde{\Delta}$ we have $\mu\Theta = strip(\mu\tilde{\Theta})$.

In the next sections we will develop some theory in which these ideas are worked out.

Section 3 Retractions

In this section we develop a little theory about pairs of cpo's, one of which can be considered less abstract than the other. We will give sufficient conditions under which the least fixed point of a transformation maps on the least fixed point of its abstract version.

(3.1) DEFINITION 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 $j \circ i = id_D$.

We write in the above situation $D \leq_{i,\,j} \tilde{D}$ or just $D \leq \tilde{D}$. If $D \leq_{i,\,j} \tilde{D}$ then i is an embedding and j is strict. (Injectivity of i follows directly from $j \circ i = id_D$; strictness of j follows from $j(\bot_{\tilde{D}}) \leq_D j(i(\bot_D)) = \bot_D$.) In the context of $D \leq_{i,\,j} \tilde{D}$ we call i the inclusion and j the retraction, respectively.

Consider the cpo's $D = \Sigma_\bot$, $\tilde{D} = \tilde{\Sigma}^{st}$ augmented with the stream ordering. Let $i : D \to \tilde{D}$ be the inclusion and let $j : \tilde{D} \to D$ be defined by $j = strip$, cf. section 2. Then we have $j \circ i(\bot) = strip(\bot) = \bot$ and $j \circ i(\sigma) = strip(\sigma.\varepsilon) = \sigma$. So $D \leq_{i,\,j} \tilde{D}$, i.e. $\Sigma_\bot \leq_{i,\,strip} \tilde{\Sigma}^{st}$.

The relation $\leq_{i,\,j}$ between cpo's is - roughly speaking - one half of the subdomain ordering in the category *CPO*. For the subdomain ordering there is the additional requirement that $i \circ j \leq id_{\tilde{D}}$. See [Pl].

Given cpo's D, $\tilde{D}$ the pair of continuous functions i, j such that $D \leq_{i,\,j} \tilde{D}$ is not unique. It is already the case (contrary to the subdomain ordering) that for fixed inclusion i there exist several retractions j such that $D \leq_{i,\,j} \tilde{D}$. For example, take $D = \tilde{D} = \mathbb{N} \cup \{ \infty \}$ together with the standard ordering. Define $i : D \to \tilde{D}$ by $i(d) = 2d$. Define $j_1, j_2 : \tilde{D} \to D$ by $j_1(\tilde{d}) = \lfloor \tilde{d}/2 \rfloor$ and $j_2(\tilde{d}) = \lceil \tilde{d}/2 \rceil$. Clearly both j_1 and j_2 are continuous and satisfy $j_1 \circ i = id_D$, $j_2 \circ i = id_D$.

(3.2) DEFINITION 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 $l \circ \tilde{\phi} = \phi \circ j$. We define the function space $\tilde{D} \sim\!> \tilde{E}$ of canonical mappings from $\tilde{D}$ to $\tilde{E}$ by $\tilde{D} \sim\!> \tilde{E} = \{ \tilde{\phi} : \tilde{D} \to \tilde{E} \mid \tilde{\phi}$ canonical $\}$.

If $\tilde{\phi} : \tilde{D} \to \tilde{E}$ is canonical, then there exists a unique $\phi : D \to E$ such that $l \circ \tilde{\phi} = \phi \circ j$. For if $\phi_1, \phi_2 : D \to E$ with $l \circ \tilde{\phi} = \phi_1 \circ j = \phi_2 \circ j$ then we have $\phi_1 = \phi_1 \circ j \circ i = l \circ \tilde{\phi} \circ i = \phi_2 \circ j \circ i = \phi_2$.

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 reformulate canonicity in terms of the equivalence relations $\sim_D$ and $\sim_E$. We will have the equivalence of (i) $\tilde{\phi} : \tilde{D} \to \tilde{E}$ is canonical and (ii) the induced mapping on equivalence classes $\tilde{\phi} : \tilde{D} / \sim_D \to \tilde{E} / \sim_E$ is well defined.

(3.3) LEMMA *Suppose* $D \leq_{i,j} \tilde{D}$, $E \leq_{k,l} \tilde{E}$. *Then it holds that* $\tilde{\phi} : \tilde{D} \to \tilde{E}$ *is canonical* $\Leftrightarrow$ $\forall d, d' \in \tilde{D}$: $d \sim_D d' \Rightarrow \tilde{\phi}(d) \sim_D \tilde{\phi}(d')$.
PROOF $''\Rightarrow''$ Choose $\phi : D \to E$ such that $l \circ \tilde{\phi} = \phi \circ j$. Let $d, d' \in \tilde{D}$ such that $d \sim_D d'$, i.e. $j(d) = j(d')$. Then we have $l(\tilde{\phi}(d)) = \phi(j(d)) = \phi(j(d')) = l(\tilde{\phi}(d'))$, so $\tilde{\phi}(d) \sim_E \tilde{\phi}(d')$.
$''\Leftarrow''$ Define $\phi : D \to E$ by $\phi = l \circ \tilde{\phi} \circ i$. Let $d \in \tilde{D}$. Then we have $\tilde{\phi}(i(j(d))) \sim_E \tilde{\phi}(d)$ and $\phi(j(d)) = l(\tilde{\phi}(i(j(d)))) = l(\tilde{\phi}(d))$, since $i(j(d)) \sim_D d$. Conclusion: $l \circ \tilde{\phi} = \phi \circ j$. □

The above lemma is not very deep but is is helpful in proving that $\tilde{D} \sim\!> \tilde{E}$ is a subcpo of the function space $\tilde{D} \to \tilde{E}$, since in the presence of 3.3 the proof takes place "in the world of $\tilde{D}$ and $\tilde{E}$."

(3.4) LEMMA *Suppose* $D \leq_{i,j} \tilde{D}$, $E \leq_{k,l} \tilde{E}$. *Then* $\tilde{D} \sim\!> \tilde{E}$ *is a cpo.*
PROOF Sufficient to prove: for a chain $\langle \phi_n \rangle_n$ in $\tilde{D} \sim\!> \tilde{E}$ is $\phi = lub_n \phi_n$ canonical. Suppose $d \sim_D d'$. Then by continuity of l and canonicity of ϕ_n: $l(\phi(d)) = l(lub_n \phi_n(d)) = lub_n l(\phi_n(d)) = lub_n l(\phi_n(d')) = l(lub_n \phi_n(d')) = l(\phi(d'))$, so $\phi(d) \sim_E \phi(d')$. □

Suppose D and E are retracts of $\tilde{D}$ and $\tilde{E}$, respectively. The function space $D \to E$ then, will be a retract of the function space $\tilde{D} \to \tilde{E}$. More precisely, if $D \leq_{i,j} \tilde{D}$ and $E \leq_{k,l} \tilde{E}$ then $(D \to E) \leq_{I,J} (\tilde{D} \to \tilde{E})$ where $I = \lambda\phi . k \circ \phi \circ j$ and $J = \lambda\tilde{\phi} . l \circ \tilde{\phi} \circ i$. Continuity of I, J follows from the continuity of i through l. Furthermore $J \circ I = \lambda\phi . l \circ k \circ \phi \circ j \circ i = \lambda\phi . id_E \circ \phi \circ id_D = id_{D \to E}$. 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 $J = \lambda\tilde{\phi} . j \circ \tilde{\phi}$.

Notice, for $\phi : D \to E$ is $I(\phi) : \tilde{D} \to \tilde{E}$ canonical: if $d \sim_D d'$ then $I(\phi)(d) = k(\phi(j(d))) = k(\phi(j(d'))) = I(\phi)(d')$ and hence $I(\phi)(d) \sim_E I(\phi)(d')$. So we have $D \to E \leq \tilde{D} \sim\!> \tilde{E}$.

Moreover I and J preserve continuity, i.e. $\phi \in [\, D \to E\,] \;\Rightarrow\; I(\phi) \in [\, \tilde{D} \to \tilde{E}\,]$ and $\tilde{\phi} \in [\, \tilde{D} \to \tilde{E}\,] \;\Rightarrow J(\tilde{\phi}) \in [\, D \to E\,]$. Therefore we have $[\, D \to E\,] \leq [\, \tilde{D} \to \tilde{E}\,]$. Combination of this two facts yields $[\, D \to E\,] \leq [\, \tilde{D} \sim\!> \tilde{E}\,]$.

The notion of a retract was introduced here with the comparison of fixed points of higher order transformations in mind. By virtue of theorem 3.6 below it would therefore be convenient to have available a means for checking canonicity of these (higher order) transformations.

(3.5) LEMMA

(i) *Choose cpo's* $D \leq_{i,j} \tilde{D}$, $E \leq_{k,l} \tilde{E}$, $F \leq_{m,n} \tilde{F}$. *Fix* $\Phi : \tilde{D} \to \tilde{E} \to \tilde{F}$. *Then it holds that* $\Phi : \tilde{D} \sim\!> \tilde{E} \sim\!> \tilde{F} \;\Leftrightarrow\; \forall d,d' \in \tilde{D},\, \forall e,e' \in \tilde{E}\colon d \sim_D d' \wedge e \sim_E e' \Rightarrow \Phi(d)(e) \sim_F \Phi(d')(e')$.

(ii) *Let V be a set, D,* $\tilde{D}$, *E,* $\tilde{E}$ *cpo's such that* $D \leq_{i,j} \tilde{D}$, $E \leq_{k,l} \tilde{E}$ *and fix* $\Phi : \tilde{D} \to V \to \tilde{E}$. *Then it holds that* $\Phi : \tilde{D} \sim\!> V \to \tilde{E} \;\Leftrightarrow\; \forall d, d' \in \tilde{D} \;\; \forall v \in V\colon \; d \sim_D d' \;\Rightarrow\; \Phi(d)(v) \sim_E \Phi(d')(v)$.

PROOF We only check (i). For $\phi, \phi' \in \tilde{E} \sim\!> \tilde{F}$ we have $\phi \sim_{E \to F} \phi' \;\Leftrightarrow\; [\, e \sim_E e' \Rightarrow \phi(e) \sim_F \phi'(e')\,]$. For it holds that $\phi \sim_{E \to F} \phi' \;\Leftrightarrow\; n \circ \phi \circ k = n \circ \phi' \circ k \;\Leftrightarrow\; \forall e \in E : n(\phi(k(e))) = n(\phi'(k(e))) \;\Leftrightarrow\; \forall e \in E : \phi(k(e)) \sim_F \phi'(k(e)) \;\Leftrightarrow\; [\, e \sim_E e' \Rightarrow \phi(e) \sim_F \phi'(e')\,]$ since $k(l(e)) \sim_E e$. □

Let ~ be the equivalence relation induced by *strip* on the several domains. We verify that $\tilde{\Phi} \in (\mathit{Conf} \to \tilde{\Sigma}^{st}) \sim\!> (\mathit{Conf} \to \tilde{\Sigma}^{st})$. Choose $O, O' \in \mathit{Conf} \to \tilde{\Sigma}^{st}$ such that $O \sim O'$ and pick $C \in \mathit{Conf}$. If $C = [\, \varepsilon, \sigma\,]$, then $\tilde{\Phi}OC = \sigma \sim \sigma = \tilde{\Phi}O'C$. If $C \to C'$ then $\tilde{\Phi}OC = \tau.O(C) \sim \tau.O'(C) = \tilde{\Phi}O'C$ since by assumption $O \sim O'$ and $x \sim y \Rightarrow \tau.x \sim \tau.y$.

Suppose $D \leq_{i,j} \tilde{D}$, $E \leq_{k,l} \tilde{E}$ and $F \leq_{m,n} \tilde{F}$. Say $E \to F \leq_{K,L} \tilde{E} \to \tilde{F}$ where $K = \lambda\psi\,.\, m \circ \psi \circ l$ and $L = \lambda\tilde{\psi}\,.\, n \circ \tilde{\psi} \circ k$. Then $D \to E \to F \leq_{I,J} \tilde{D} \to \tilde{E} \to \tilde{F}$ where $I\phi\tilde{d}\tilde{e} = K(\phi(j\tilde{d}))\tilde{e} = m(\phi(j\tilde{d})(l\tilde{e}))$ and $J\tilde{\phi}de = L(\tilde{\phi}(id))e = n(\tilde{\phi}(id)(ke))$. Slightly more general we have, if $D_\alpha \leq_{i_\alpha, j_\alpha} \tilde{D}_\alpha$ for $\alpha \in \{1, .., n\}$ and $E \leq_{k,l} \tilde{E}$ then $D_1 \to .. \to D_n \to E \leq_{I,J} \tilde{D}_1 \to .. \to \tilde{D}_n \to \tilde{E}$ where $I\Phi\tilde{d}_1 .. \tilde{d}_n = k(\Phi(j_1\tilde{d}_1) .. (j_n\tilde{d}_n))$ and $J\tilde{\Phi}d_1 .. d_n = l(\tilde{\Phi}(i_1 d_1) .. (i_n d_n))$. Similarly it is possible to extend lemma 3.5: $\Phi\colon (\tilde{D}_1 \to .. \to \tilde{D}_n \to \tilde{E}) \to (\tilde{D}_1 \to .. \to \tilde{D}_n \to \tilde{E})$ we have $\Phi \in (\tilde{D}_1 \sim\!> .. \sim\!> \tilde{D}_n \sim\!> \tilde{E}) \sim\!> (\tilde{D}_1 \sim\!> .. \sim\!> \tilde{D}_n \sim\!> \tilde{E}) \Leftrightarrow d_1 \sim_1 d_1', .., d_n \sim_n d_n' \Rightarrow \Phi d_1 .. d_n \sim_E \Phi d_1' .. d_n'$. We will use this unraveling of the notion of canonicity in section 5.

Finally in this section we arrive at the theorem that relates least fixed points of a transformation in the function space $\tilde{D} \to \tilde{D}$ to the least fixed point of its retract in the function space $D \to D$. This theorem is strongly related to the Fixed Point Transformation

Lemma. (See [BMZ]a, [Me]a.)

(3.6) THEOREM *Suppose* $D \leq_{i,j} \tilde{D}$. *Let* $\tilde{\phi} : \tilde{D} \to \tilde{D}$ *be continuous and canonical. Put* $J(\tilde{\phi}) = \phi$. *Then* $\phi : D \to D$ *is continuous with* $\mu\phi = j(\mu\tilde{\phi})$.
PROOF Clearly, ϕ is continuous by definition of J. By canonicity of $\tilde{\phi}$ we have $\phi \circ j = j \circ \tilde{\phi}$: $\phi(j(d)) = J(\tilde{\phi})(j(d)) = j(\tilde{\phi}(i(j(d)))) = j(\tilde{\phi}(d))$ since $i(j(d)) \sim_D d$.

By induction on n we derive $j(\tilde{\phi}^n(\bot_{\tilde{D}})) = \phi^n(\bot_D)$. Basis, n=0: Directly from the strictness of j. Induction step, $n > 0$: $j(\tilde{\phi}^n(\bot_{\tilde{D}})) = j(\tilde{\phi}(\tilde{\phi}^{n-1}(\bot_{\tilde{D}}))) = \phi(j(\tilde{\phi}^{n-1}(\bot_{\tilde{D}}))) = \phi(\phi^{n-1}(\bot_D)) = \phi^n(\bot_D)$ by the equality $j \circ \tilde{\phi} = \phi \circ j$ and the induction hypothesis. By continuity of j we conclude: $j(\mu\tilde{\phi}) = j(\ lub_n\ \tilde{\phi}^n(\bot_{\tilde{D}})\ ) = lub_n\ j(\tilde{\phi}^n(\bot_{\tilde{D}})) = lub_n\ \phi^n(\bot_D) = \mu\phi$. □

Section 4 Operational Semantics and Denotational Semantics for $\mathcal{B}$

In this section we introduce the abstract backtracking language $\mathcal{B}$. This uniform language was studied also in [BV]a for it captures the control flow of Prolog with cut, the latter being the main interest of the particular paper. (See also [BaKo], [Vi]a, [Ba2]a for similar uses of intermediate abstracta in deriving sound denotational semantics for logic programming languages.) In the present paper however, we will focus on the residue $\mathcal{B}$ itself to serve as a case study for our method of comparing operational and denotational semantics.

(4.1) DEFINITION Fix a set of actions *Action* and a set of procedure names *Proc*. We define the set of elementary statements $EStat = \{ a, \textbf{\textit{fail}}, \textbf{!}, s_1\ \textbf{\textit{or}}\ s_2, x \mid a \in Action, s_i \in Stat, x \in Proc \}$, the set of statements $Stat = \{ e_1 : .. : e_r \mid r \in \mathbb{N}, e_i \in EStat \}$ and the set of declarations $Decl = \{ x_1 \leftarrow s_1 : .. : x_r \leftarrow s_r \mid r \in \mathbb{N}, x_i \in Proc, s_i \in Stat, i \neq j \Rightarrow x_i \neq x_j \}$. The backtracking language $\mathcal{B}$ is defined by $\mathcal{B} = \{ d \mid s \mid d \in Decl, s \in Stat \}$.

So a $\mathcal{B}$ program is a declaration together with a statement. Such a statement is a -possibly empty - list of elementary statements of one of the formats action *a*, procedure variable *x*, explicit failure ***fail***, cut operator **!** and alternative composition $s_1\ \textbf{\textit{or}}\ s_2$.

We let *a* range over *Action*, *x* over *Proc*, *e* over *EStat*, *s* over *Stat* and *d* over *Decl*. We write $x \leftarrow s \in d$ if $x \leftarrow s = x_i \leftarrow s_i$ (for some i) or if $s = \textbf{\textit{fail}}$ otherwise. (By this convention we do not have free procedure variables in a statement, since every *x* is declared in *d* having by default the procedure body ***fail***.)

(4.2) DEFINITION Fix a set Σ of states. Define the set of generalized statements by $GStat = \{ \langle s_1, D_1 \rangle : .. : \langle s_r, D_r \rangle \mid r \in \mathbb{N}, s_i \in Stat, D_i \in Stack \}$. Let γ denote the empty generalized statement. Define the set of frames by $Frame = \{ [g, \sigma] \mid g \in GStat, \sigma \in \Sigma \}$ and the set of

stacks by $Stack = \{ F_1 : .. F_r \mid r \in \mathbb{N}, F_i \in Frame \}$. We use E to denote the empty stack.

Next we describe the operational meaning for $\mathcal{B}$. Consider the program $d \mid s$ and a state σ. The declaration d induces a transition system (also called d). The meaning $\mathcal{O}[\![d \mid s]\!] \sigma$ then will be the stream of labels of the computation with respect to the transition system d starting from an initial configuration associated with s and σ.

We introduce the collection of Σ-transition systems TS by $TS = Stack \rightarrow_{part} (Stack \cup \Sigma \times Stack)$. For $t \in TS$ we shall write $S \rightarrow_t S'$ if $t(S) = S' \in Stack$ and $S \rightarrow_t^\sigma S'$ if $t(S) = (\sigma, S') \in \Sigma \times Stack$. We fix an action interpretation $I : Action \rightarrow \Sigma \rightarrow_{part} \Sigma$, that reflects the effect of the execution of an action on a state. (The language $\mathcal{B}$ gains flexibility if actions are allowed to succeed in one state, while failing in another.)

(4.3) DEFINITION Let $d \in Decl$. d induces a transition system d in TS which is defined as the smallest element of TS (with respect to $\subseteq$) such that

(i) $[\gamma, \sigma] : S \rightarrow_d^\sigma S$

(ii) $[\langle \varepsilon , D \rangle : g , \sigma] : S \rightarrow_d [g , \sigma] : S$

(iii) $[\langle a ; s , D \rangle : g , \sigma] : S \rightarrow_d [\langle s , D \rangle : g , \sigma'] : S$ if $\sigma' = I(a)(\sigma)$ exists

$[\langle a ; s , D \rangle : g , \sigma] : S \rightarrow_d S$ otherwise

(iv) $[\langle \textbf{\textit{fail}} ; s , D \rangle : g , \sigma] : S \rightarrow_d S$

(v) $[\langle \textbf{!} ; s , D \rangle : g , \sigma] : S \rightarrow_d [\langle s , D \rangle : g , \sigma] : D$

(vi) $[\langle x' ; s , D \rangle : g , \sigma] : S \rightarrow_d [\langle s' , S \rangle : \langle s , D \rangle : g , \sigma] : S$ where $x' \leftarrow s' \in d$

(vii) $[\langle (s_1 \textbf{ or } s_2) ; s , D \rangle : g , \sigma] : S \rightarrow_d F_1 : F_2 : S$

where $F_i = [\langle s_i ; s , D \rangle : g , \sigma]$ $(i = 1, 2)$

A stack $S \in Stack$ is a stack of alternatives. Each alternative, i.e. each frame, can be thought of as holding a (partial) elaboration of an initial statement-state pair, also referred to as the original goal. The top frame on the stack is the alternative to be tried first. There is no transition specified for the empty stack.

If the top frame F holds no proper statements, i.e. $F = [\gamma, \sigma]$, the state σ is outputted on the transition, since the initial goal has been solved yielding σ, and the computation continues with the alternatives embodied by the remainder of the stack. (For we want to deliver all the answers for the initial goal.) If the top frame does contain a proper statement, say $F = [\langle s , D \rangle : g, \sigma]$, an internal transition is made, that depends on the structure of s. The empty component $\langle \varepsilon , D \rangle$ is just skipped.

In case of $a ; s$ the action interpretation I is consulted for the result of action a in state σ. If a transforms σ successfully into a new state σ', the state of the frame F is changed accordingly and the computation continues with the statement s in F. If a can not be executed successfully in state σ, i.e. $Ia\sigma$ is not defined, this will be a failure for the whole frame F: the

alternative is pushed of the stack and the computation continues with the alternatives left on the failure stack *S*. An explicit ***fail*** is handled similarly.

A cut can always be executed with success. But, there is a side effect. To implement this side-effect we make use of the cut information represented by the dump stack associated with a statement. This dump stack contains the alternatives that were open at the moment the statement was introduced. Executing a cut means restoring these alternatives which amounts to removal of the alternatives that were created after this (occurrence of) **!** was introduced. So in the right-hand side the failure stack *S* will be replaced by the dump stack *D*.

In case of a procedure call we apply body replacement. Thus we introduce a new statement, viz. s' in the top frame. Since *S* consists of the alternatives that are open at this creation time of s' we attach to s' the stack *S* as its dump stack. In case of an alternative composition s_1 ***or*** s_2 the top frame splits into two frames. The uppermost corresponds to s_1, the other is associated with s_2. So the alternative induced by s_1 will be tried first.

We will associate with a declaration *d* and its induced transition system $\rightarrow_d$ an answer function α_d: $Stack \rightarrow \Sigma^{st}$ that for stacks *S* yields the concatenation of the σ-labels of the computation starting from *C* according to the transition system *d*. We use a higher-order transformation Φ_d for a fixed point definition of α_d.

(4.4) DEFINITION Let $d \in Decl$. Define $\Phi_d \in [\,(Stack \rightarrow \Sigma^{st}) \rightarrow (Stack \rightarrow \Sigma^{st})\,]$ by $\Phi_d(\alpha)(E) = \varepsilon$, $\Phi_d(\alpha)(S) = \alpha(S')$ if $S \rightarrow_d S'$, $\Phi_d(\alpha)(S) = \sigma \bullet \alpha(S')$ if $S \rightarrow_d^{\sigma} S'$. The answer function α_d: $Stack \rightarrow \Sigma^{st}$ associated with the Σ-transition system *d* is defined by $\alpha_d = \mu\Phi_d$.

It is straightforward to check that Φ_d is well-defined and thus that its least fixed point exists. This answer function is used to formulate the operational semantics for $\mathcal{B}$.

(4.5) DEFINITION The operational semantics $\mathcal{O}$: $\mathcal{B} \rightarrow \Sigma \rightarrow \Sigma^{st}$ for the backtracking language $\mathcal{B}$ is defined by $\mathcal{O}(d\,|\,s)(\sigma) = \alpha_d([\,\langle s, E\rangle, \sigma\,])$ where α_d is the answer function associated with *d*.

Next we define a denotational semantics for $\mathcal{B}$.

(4.6) DEFINITION

(i) We define the set of failure continuations $FCont = \Sigma^{st}$, the set of cut continuations $CCont = \Sigma^{st}$, the set of success continuations $SCont = [\,FCont \rightarrow [\,CCont \rightarrow \Sigma \rightarrow \Sigma^{st}\,]\,]$, the set of meanings $Meaning = Stat \rightarrow [\,SCont \rightarrow [\,FCont \rightarrow [\,CCont \rightarrow \Sigma \rightarrow \Sigma^{st}\,]\,]\,]$. We denote by σ, ϕ, κ, ξ and *M* typical elements of Σ, *FCont*, *CCont*, *SCont* and *Meaning*, respectively.

(ii) The denotational semantics $\mathcal{D}$: $\mathcal{B} \rightarrow \Sigma \rightarrow \Sigma^{st}$ for the backtracking language $\mathcal{B}$ is defined

by $\mathcal{D}(d \mid s)(\sigma) = M_d[\![s]\!] \xi_o \phi_o \kappa_o \sigma$ where $\xi_o = \lambda\phi\kappa\sigma.\sigma \bullet \phi$ and $\phi_o = \kappa_o = \varepsilon$. Here M_d is the least fixed point of $\Psi_d \in [\, \mathit{Meaning} \to \mathit{Meaning}\,]$ defined by

$\Psi_d M[\![\varepsilon]\!] \xi\phi\kappa\sigma = \xi\phi\kappa\sigma$

$\Psi_d M[\![a]\!] \xi\phi\kappa\sigma = \xi\phi\kappa\sigma'$ if $\sigma' = I(a)(\sigma)$ exists

$\Psi_d M[\![a]\!] \xi\phi\kappa\sigma = \phi$ otherwise

$\Psi_d M[\![\textbf{\textit{fail}}]\!] \xi\phi\kappa\sigma = \phi$

$\Psi_d M[\![!]\!] \xi\phi\kappa\sigma = \xi\kappa\kappa\sigma$

$\Psi_d M[\![s_1 \textbf{\textit{or}} s_2]\!] \xi\phi\kappa\sigma = M[\![s_1]\!] \xi\{M[\![s_2]\!] \xi\phi\kappa\sigma\}\kappa\sigma$

$\Psi_d M[\![x]\!] \xi\phi\kappa\sigma = M[\![s]\!] \{\lambda\bar{\phi}\bar{\kappa}.\xi\bar{\phi}\kappa\}\phi\phi\sigma$ where $x \leftarrow s \in d$

$\Psi_d M[\![e; s]\!] \xi\phi\kappa\sigma = M[\![e]\!] \{M[\![s]\!] \xi\}\phi\kappa\sigma$

We leave it to the reader to verify the well-definedness of Ψ but comment briefly on the intuition behind the clauses above.

The transformation is triggered by the statement *s*. In case of an empty statement we consider the initial statement to be executed successfully. So the success continuation is applied to the particular arguments. In case of a primitive action *a* that transforms the state σ into the state σ' we also apply the success continuation but now to the new state σ'. If *a* fails in state σ we deliver the failure continuation ϕ as a denotation. Analogously for the explicit ***fail***. A cut operator can always be executed successfully but as a side effect the failure continuation is replaced by the cut continuation. For the alternative composition we evaluate the first alternative s_1 according to the meaning *M* and add the other alternative s_2 on top of the failure continuation. Procedure calls are handled by means of body replacement. The several continuations are changed appropriately. A sequential composition is denoted by the meaning of its first elementary statement while pushing the remainder into the success continuation.

The denotational semantics for $\mathcal{B}$ can be computed given a program $d \mid s$ from the least fixed point M_d of the transformation Ψ_d using so called standard continuations. Note the format of the standard success continuation $\xi_o = \lambda\phi\kappa\sigma.\sigma \bullet \phi$. This will amount to delivering all remaining alternatives after the first solution is computed.

Section 5 Relating $\mathcal{O}$ and $\mathcal{D}$

In this section we will relate the operational and denotational semantics for $\mathcal{B}$ of the previous section. This will be done similarly to the case of the simple while language of section 2: We extend the defining transformations Φ and Ψ to less abstract transformations $\tilde{\Phi}$ and $\tilde{\Psi}$. Using the results on retracts we infer from the equivalence of $\tilde{\Phi}$ and (an variant of) $\tilde{\Psi}$ the equivalence of $\mathcal{O}$ and $\mathcal{D}$.

Recall from section 2 the definition of the cpo $\tilde{\Sigma}^{st}$ of streams over $\tilde{\Sigma}$. By Σ^{st} we denote the subcpo of $\tilde{\Sigma}^{st}$ induced by $\Sigma^* \cup \Sigma^*.\bot \cup \Sigma^\omega$.

(5.1) DEFINITION The function $strip : \tilde{\Sigma}^{st} \to \Sigma^{st}$ is defined by $strip = \mu P$ where $P \in [\, [\, \tilde{\Sigma}^{st} \to \Sigma^{st}\,] \to [\, \tilde{\Sigma}^{st} \to \Sigma^{st}\,]\,]$ is defined by $P\rho\bot = \bot$, $P\rho\varepsilon = \varepsilon$, $P\rho\sigma.\, x = \sigma.\rho x$, $P\rho\tau.\, x = \rho x$.

So *strip* substitutes ε for finitely many τ's and $\bot$ for ω many. Since the operator P that defines *strip* is continuous we can easily check by means of fixed point induction the distributivity of *strip* over $\bullet$, i.e. $strip(x \bullet y) = strip(x) \bullet strip(y)$.

(5.2) LEMMA Σ^{st} *is a retract of* $\tilde{\Sigma}^{st}$.
PROOF By continuity of *strip* and the inclusion mapping $incl: \Sigma^{st} \to \tilde{\Sigma}^{st}$ it suffices to show $strip \circ incl = id_{\Sigma^{st}}$, i.e. $\forall x \in \Sigma^{st}: strip(x) = x$. It is straightforward to show by induction on n: $(*)$ $\forall n \in \mathbb{N}\, \forall x \in \Sigma^n \cup \Sigma^n.\bot: strip(x) = x$. Now choose $x \in \Sigma^{st}$ arbitrary. Let $\langle x_n \rangle_n$ be a chain in $\Sigma^* \cup \Sigma^*.\bot$ with least upperbound x. Then we have by continuity of *strip* and by $(*)$: $strip(x) = lub_n\, strip(x_n) = lub_n\, x_n = x$. □

All retractions considered in the remainder of this section will be derived from *strip* and *incl* using the construction for function spaces as described after lemma 3.4. From now on retractions will be denoted by *I*, *J* (but also by *strip*).

We continue with the extension of the operational semantics. Now for all transitions we will have a label from $\tilde{\Sigma}$. But except for this, definitions 4.3 and 5.3 are the same. So for example, we again make use of the action interpretation $I : Action \to \Sigma \to_{part} \Sigma$ to establish the behavior of an action a in a state σ. Furthermore, we let $\tilde{TS}$ denote the collection of $\tilde{\Sigma}$-transition systems $Stack \to_{part} \tilde{\Sigma} \times Stack$. We use similar conventions as for Σ-transition systems.

(5.3) DEFINITION Let $d \in Decl$. d induces a transition system d in $\tilde{TS}$ which is defined as the smallest element of $\tilde{TS}$ (with respect to $\subseteq$) such that

(i) $[\, \gamma, \sigma\,] : S \to_d^\sigma S$

(ii) $[\, \langle \varepsilon, D\rangle : g, \sigma\,] : S \to_d^\tau [\, g, \sigma\,] : S$

(iii) $[\, \langle a; s, D\rangle : g, \sigma\,] : S \to_d^\tau [\, \langle s, D\rangle : g, \sigma'\,] : S$ if $\sigma' = I(a)(\sigma)$ exists
$[\, \langle a; s, D\rangle : g, \sigma\,] : S \to_d^\tau S$ otherwise

(iv) $[\, \langle \textbf{\textit{fail}}; s, D\rangle : g, \sigma\,] : S \to_d^\tau S$

(v) $[\, \langle \textbf{!}; s, D\rangle : g, \sigma\,] : S \to_d^\tau [\, \langle s, D\rangle : g, \sigma\,] : D$

(vi) $[\, \langle x'; s, D\rangle : g, \sigma\,] : S \to_d^\tau [\, \langle s', S\rangle : \langle s, D\rangle : g, \sigma\,] : S$ where $x' \leftarrow s' \in d$

(vii) $[\,\langle (s_1 \textbf{ or } s_2)\,; s\,, D\rangle{:}g\,, \sigma\,] : S \;\to_d^{\tau}\; F_1 : F_2 : S$ where $F_i = [\,\langle s_i\,; s\,, D\rangle{:}g\,, \sigma\,]$ $(i = 1, 2)$

We shall associate with a declaration d an answer function $\tilde{\alpha}_d$: $Stack \to \tilde{\Sigma}^{st}$ that for stacks S yields the concatenation of all the labels of the computation starting from S according to the transition system d. As before we use a higher-order transformation $\tilde{\Phi}_d$ for a fixed point definition of $\tilde{\alpha}_d$. Note that we presently also demand canonicity for the transformation $\tilde{\Phi}_d$.

(5.4) DEFINITION Let $d \in Decl$. Define $\tilde{\Phi}_d \in [\,(Stack \to \tilde{\Sigma}^{st}) \sim\!> (Stack \to \tilde{\Sigma}^{st})\,]$ by $\tilde{\Phi}_d(\alpha)(E) = \varepsilon$, $\tilde{\Phi}_d(\alpha)(S) = \theta \bullet \alpha(S')$ if $S \to_d^{\theta} S'$. The answer function $\tilde{\alpha}_d$: $Stack \to \tilde{\Sigma}^{st}$ associated with the $\tilde{\Sigma}$-transition system d is defined by $\tilde{\alpha}_d = \mu\tilde{\Phi}_d$.

Canonicity of $\tilde{\Phi}_d$ (as well as continuity) is straightforward to check: Let $\alpha, \alpha' \in Stack \to \tilde{\Sigma}^{st}$ such that $\alpha \sim \alpha'$. We have to show: $\tilde{\Phi}_d(\alpha) \sim \tilde{\Phi}_d(\alpha')$, i.e. $\forall S \in Stack$: $strip(\tilde{\Phi}_d(\alpha)(S)) = strip(\tilde{\Phi}_d(\alpha')(S))$. Let $S \in Stack$. Without loss of generality $S \neq E$. Say $S \to_d^{\theta} S'$. Then we have $strip(\tilde{\Phi}_d(\alpha)(S)) = strip(\theta \bullet \alpha(S')) = strip(\theta) \bullet strip(\alpha(S')) = strip(\theta) \bullet strip(\alpha'(S')) = strip(\theta \bullet \alpha'(S')) = strip(\tilde{\Phi}_d(\alpha')(S))$ since $\alpha(S') \sim \alpha'(S')$ by assumption.

The pleasant property of the new transformation $\tilde{\Phi}$, as was elaborated upon before, is the uniqueness of its least fixed point.

(5.5) LEMMA *For all $d \in Decl$: $\tilde{\Phi}_d$ has a unique fixed point.*
PROOF Let $d \in Decl$. Uniqueness of the least fixed point of $\tilde{\Phi}_d$, which exists by continuity of $\tilde{\Phi}_d$, follows from the fact that $\forall S \in Stack$, $\tilde{\alpha}_d(S) \in \tilde{\Sigma}^* \cup \tilde{\Sigma}^{\omega}$. For this property implies maximality of $\mu\tilde{\Phi}_d$: Let $S \in \mathcal{S} = \{\, \bar{S} \in Stack \mid \tilde{\alpha}_d(\bar{S}) \in \tilde{\Sigma}^*.\bot \,\}$ be of minimal length. Then $S \neq E$, so $S \to_d^{\theta} S'$ for some $\theta \in \tilde{\Sigma}$, $S' \in Stack$. But then $\tilde{\alpha}_d(S') \in \tilde{\Sigma}^*.\bot$ is of length strictly less than $\tilde{\alpha}_d(S)$. Conclusion: $\mathcal{S}$ is empty, so $\forall S \in Stack$: $\tilde{\alpha}_d(S) \in \tilde{\Sigma}^* \cup \tilde{\Sigma}^{\omega}$. □

Next we check that the new answer function $\tilde{\alpha}_d$ derived from $\tilde{\Psi}_d$ equals the old answer function α_d derived from Ψ_d modulo clock ticks τ.

(5.6) LEMMA *For $d \in Decl$, $strip(\tilde{\alpha}_d) = \alpha_d$.*
PROOF Let $d \in Decl$. By theorem 3.6 it suffices to show: $strip(\tilde{\Phi}_d) = \Phi_d$. This is clear, since $\forall \alpha \in Stack \to \Sigma^{st}$, $S \in Stack$: $J(\tilde{\Phi}_d)(\alpha)(S) = strip(\tilde{\Phi}_d(I(\alpha))(S)) = strip(\varepsilon) = \varepsilon = \Phi_d(\alpha)(S)$ if $S = E$, and $J(\tilde{\Phi}_d)(\alpha)(S) = strip(\tilde{\Phi}_d(I(\alpha))(S)) = strip(\theta \bullet I(\alpha)(S')) = strip(\theta) \bullet strip(I(\alpha)(S')) = \Phi_d(\alpha)(S)$ if $S \to_d^{\theta} S'$. □

Next we formulate the extension of the higher order transformation Ψ. Note that we restrict not to "continuous" continuations but rather to both "continuous and canonical" ones.

(5.7) DEFINITION

(i) We define the set of failure continuations $FCont^{\sim} = \tilde{\Sigma}^{st}$, the set of cut continuations $CCont^{\sim} = \tilde{\Sigma}^{st}$, the set of success continuations $SCont^{\sim} = [\, FCont^{\sim} \sim\!> [\, CCont^{\sim} \sim\!> \Sigma \to \tilde{\Sigma}^{st} \,]\,]$, the set of meanings $Meaning^{\sim} = Stat \to [\, SCont^{\sim} \sim\!> [\, FCont^{\sim} \sim\!> [\, CCont^{\sim} \sim\!> \Sigma \to \tilde{\Sigma}^{st} \,]\,]\,]$. We denote by σ, ϕ, κ, ξ and M typical elements of $\tilde{\Sigma}$, $FCont^{\sim}$, $CCont^{\sim}$, $SCont^{\sim}$ and $Meaning^{\sim}$, respectively.

(ii) Let $d \in Decl$. By $\tilde{M}_d$ we denote the least fixed point of $\tilde{\Psi}_d \in [\, Meaning^{\sim} \sim\!> Meaning^{\sim}]$ defined by

$$\tilde{\Psi}_d M[\![\, \varepsilon \,]\!]\, \xi\phi\kappa\sigma = \tau \bullet \xi\phi\kappa\sigma$$
$$\tilde{\Psi}_d M[\![\, a \,]\!]\, \xi\phi\kappa\sigma = \tau \bullet \xi\phi\kappa\sigma' \quad \text{if } \sigma' = I(a)(\sigma) \text{ exists}$$
$$\tilde{\Psi}_d M[\![\, a \,]\!]\, \xi\phi\kappa\sigma = \tau \bullet \phi \quad \text{otherwise}$$
$$\tilde{\Psi}_d M[\![\, \textbf{\textit{fail}} \,]\!]\, \xi\phi\kappa\sigma = \tau \bullet \phi$$
$$\tilde{\Psi}_d M[\![\, ! \,]\!]\, \xi\phi\kappa\sigma = \tau \bullet \xi\kappa\kappa\sigma$$
$$\tilde{\Psi}_d M[\![\, s_1 \textbf{\textit{ or }} s_2 \,]\!]\, \xi\phi\kappa\sigma = \tau \bullet M[\![\, s_1 \,]\!]\, \xi\{M[\![\, s_2 \,]\!]\, \xi\phi\kappa\sigma\}\kappa\sigma$$
$$\tilde{\Psi}_d M[\![\, x \,]\!]\, \xi\phi\kappa\sigma = \tau \bullet M[\![\, s \,]\!]\, \{\lambda\bar{\phi}\bar{\kappa}.\xi\bar{\phi}\kappa\}\phi\phi\sigma \quad \text{where } x \leftarrow s \in d$$
$$\tilde{\Psi}_d M[\![\, e; s \,]\!]\, \xi\phi\kappa\sigma = M[\![\, e \,]\!]\, \{M[\![\, s \,]\!]\, \xi\}\phi\kappa\sigma$$

Again it is noteworthy that definitions 4.6 and 5.7 are the same except for occurrences of τ.

It is a matter of routine to check $\forall M \in Meaning^{\sim}$: $\tilde{\Psi}_d M \in Stat \to [\, SCont^{\sim} \to [\, FCont^{\sim} \to [\, CCont^{\sim} \to \Sigma \to \tilde{\Sigma}^{st} \,]\,]\,]$ and that moreover $\forall M, M' \in Meaning^{\sim}$ such that $M \sim M'$, $\forall s \in Stat$, $\forall \xi, \xi' \in SCont^{\sim}$ such that $\xi \sim \xi'$, $\forall \phi, \phi' \in FCont^{\sim}$ such that $\phi \sim \phi'$, $\forall \kappa, \kappa' \in CCont^{\sim}$ such that $\kappa \sim \kappa'$, $\forall \sigma \in \Sigma$: $\tilde{\Psi}_d M s \xi\phi\kappa\sigma \sim \tilde{\Psi}_d M' s \xi'\phi'\kappa'\sigma$. So by 3.5 $\tilde{\Psi}_d$ is well-defined.

(5.8) LEMMA *For all $d \in Decl$ we have $strip(\tilde{M}_d) = M_d$.*

PROOF Let $d \in Decl$. By theorem 3.6 it suffices to show $J(\tilde{\Psi}_d) = \Psi_d$, i.e. for $M \in Meaning$, $s \in Stat$, $\xi \in SCont$, $\phi \in FCont$, $\kappa \in CCont$, $\sigma \in \Sigma$ it holds that $J\tilde{\Psi}_d M[\![\, s \,]\!]\, \xi\phi\kappa\sigma = \Psi_d M[\![\, s \,]\!]\, \xi\phi\kappa\sigma$. This can be done by a straightforward calculation (relying heavily on the remark at the end of section 3) of which we shall exhibit only a typical case where $s = x'$.

Say $x' \leftarrow s' \in d$. $J\tilde{\Psi}_d M[\![\, x' \,]\!]\, \xi\phi\kappa\sigma = J(\tilde{\Psi}_d(IM)[\![\, x' \,]\!]\, (I\xi)(I\phi)(I\kappa)\sigma) = J(\tau \bullet (IM)[\![\, s' \,]\!]\, \{\lambda\bar{\phi}\bar{\kappa}.(I\xi)\bar{\phi}(I\kappa)\}(I\phi)(I\phi)\sigma) = J(\tau \bullet (IM)[\![\, s' \,]\!]\, \{I(\lambda\phi'\kappa'.\xi\phi'\kappa)\}(I\phi)(I\phi)\sigma) = J(\tau) \bullet JI(M[\![\, s' \,]\!]\, \{\lambda\phi'\kappa'.\xi\phi'\kappa\}\phi\phi\sigma) = M[\![\, s' \,]\!]\, \{\lambda\phi'\kappa'.\xi\phi'\kappa\}\phi\phi\sigma = \Psi_d[\![\, x \,]\!]\, \xi\phi\kappa\sigma$. Here we have used $\lambda\bar{\phi}\bar{\kappa}.(I\xi)\bar{\phi}(I\kappa) = \lambda\bar{\phi}\bar{\kappa}.I(\xi(J\bar{\phi})(JI\kappa)) = \lambda\bar{\phi}\bar{\kappa}.I(\xi(J\bar{\phi})(\kappa)) = I(\lambda\phi'\kappa'.\xi\phi'\kappa)$ and $(IM)[\![\, s \,]\!]\, (I\xi)(I\phi)(I\kappa)\sigma = I(M[\![\, s \,]\!]\, (JI\xi)(JI\phi)(JI\kappa)\sigma = I(M[\![\, s \,]\!]\, \xi\phi\kappa\sigma)$. □

The last step towards the equivalence theorem below is the formulation of the intermediate function $\tilde{\mathcal{I}}$ defined on configurations which extends $\tilde{M}_d$.

(5.9) DEFINITION Let $d \in Decl$. The mappings $\tilde{\mathcal{I}}_d$: $Conf \to \tilde{\Sigma}^{st}$, $Frame \to FCont\tilde{} \to \tilde{\Sigma}^{st}$, $GStat \to FCont\tilde{} \to \Sigma \to \tilde{\Sigma}^{st}$ are defined as follows: $\tilde{\mathcal{I}}_d[\![E]\!] = \varepsilon$; $\tilde{\mathcal{I}}_d[\![F : S]\!] = \tilde{\mathcal{I}}_d[\![F]\!] \{ \tilde{\mathcal{I}}_d[\![S]\!] \}$; $\tilde{\mathcal{I}}_d[\![[g,\sigma]\!] \phi = \tilde{\mathcal{I}}_d[\![g]\!] \phi\sigma$; $\tilde{\mathcal{I}}_d[\![\gamma]\!] \phi\sigma = \sigma \bullet \phi$; $\tilde{\mathcal{I}}_d[\![[\langle s,D \rangle : g]\!] \phi\sigma = \tilde{M}_d[\![s]\!] \{ \lambda\phi\kappa.\tilde{\mathcal{I}}_d[\![g]\!] \phi \} \phi \{ \tilde{\mathcal{I}}_d[\![D]\!] \} \sigma$.

We leave it to the reader to check the well-definedness of $\tilde{\mathcal{I}}_d$. We will check that $\tilde{\mathcal{I}}_d$ is a fixed point of the transformation $\tilde{\Phi}_d$. Therefore by lemma 5.5 we have that $\tilde{\mathcal{I}}_d$ and $\tilde{\alpha}_d$ coincide.

(5.10) LEMMA *For $d \in Decl$ we have $\tilde{\Phi}_d(\tilde{\mathcal{I}}_d) = \tilde{\mathcal{I}}_d$.*
PROOF Let $d \in Decl$. We have to verify $\tilde{\Phi}_d(\tilde{\mathcal{I}}_d)[\![S]\!] = \tilde{\mathcal{I}}_d[\![S]\!]$ for each stack S. We only treat the case $[\langle x';s,D \rangle : g,\sigma] : S$ leaving the other (similar and easier) cases to the reader.

$\tilde{\mathcal{I}}_d[\![[\langle x';s,D \rangle : g,\sigma] : S]\!] = \tilde{M}_d[\![x']\!] \{ \tilde{M}_d[\![s]\!] \xi \} \{ \tilde{\mathcal{I}}_d S \} \{ \tilde{\mathcal{I}}_d D \} \sigma =$
$\tau \bullet \tilde{M}_d[\![s']\!] \{ \lambda\phi\kappa.\tilde{M}_d[\![s]\!] \xi\phi \{ \tilde{\mathcal{I}}_d D \} \} \{ \tilde{\mathcal{I}}_d S \} \{ \tilde{\mathcal{I}}_d S \} \sigma =$
$\tau \bullet \tilde{M}_d[\![s']\!] \{ \lambda\phi\kappa.\tilde{\mathcal{I}}_d[\![[\langle s,D \rangle : g]\!] \{ \tilde{\mathcal{I}}_d S \} \} \{ \tilde{\mathcal{I}}_d S \} \{ \tilde{\mathcal{I}}_d S \} \sigma =$
$\tau \bullet \tilde{\mathcal{I}}_d[\![[\langle s',S \rangle : \langle s,D \rangle : g,\sigma] : S]\!] = \tilde{\Phi}_d \tilde{\mathcal{I}}_d[\![[\langle x';s,D \rangle : g,\sigma] : S]\!]$,
where $\xi = \lambda\phi\kappa.\tilde{\mathcal{I}}_d[\![g]\!] \{ \tilde{\mathcal{I}}_d S \}$. □

Finally we have arrived in a position in which we are able to compare the operational and denotational semantics for the abstract backtracking language $\mathcal{B}$.

(5.11) THEOREM $\mathcal{O} = \mathcal{D}$
PROOF Let $d \mid s \in \mathcal{B}$. By uniqueness of the fixed point of $\tilde{\Phi}_d$ and the above lemma we have $\tilde{\mathcal{I}}_d = \tilde{\alpha}_d$. So it follows that $\tilde{\alpha}_d[\![[\langle s,E \rangle,\sigma]]\!] = \tilde{\mathcal{I}}_d[\![[\langle s,E \rangle,\sigma]]\!] = \tilde{M}_d[\![s]\!] \{ \lambda\phi\kappa.\tilde{\mathcal{I}}_d[\![\gamma]\!] \phi \} \{ \tilde{\mathcal{I}}_d E \} \{ \tilde{\mathcal{I}}_d E \} \sigma = \tilde{M}_d[\![s]\!] \{ \lambda\phi\kappa.\sigma \bullet \phi \} \varepsilon\varepsilon\sigma = \tilde{M}_d[\![s]\!] \xi_o \phi_o \kappa_o \sigma$. Finally by the lemmas 5.6 and 5.8 we arrive at $\mathcal{O}[\![d \mid s]\!] \sigma = \alpha_d[\![[\langle s,E \rangle,\sigma]]\!] = strip(\tilde{\alpha}_d[\![[\langle s,E \rangle,\sigma]]\!]) = strip(\tilde{M}_d[\![s]\!] \xi_o \phi_o \kappa_o \sigma) = M_d[\![s]\!] \xi_o \phi_o \kappa_o \sigma = \mathcal{D}[\![d \mid s]\!] \sigma$. □

References

[BV] A. de Bruin and E.P. de Vink, ‘‘Continuation Semantics for Prolog with Cut,’’ pp. 178-192 in *Proc. TAPSOFT’89, volume 1*, J. Díaz & F. Orejas (eds.), LNCS **351** (1989).

[Ba1] J.W. de Bakker, *Mathematical Theory of Program Correctness*, Prentice Hall International, London (1980).

[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** (1988).

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

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

[BeKl] 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).

[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).

[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).

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

[BZ] J.W. de Bakker and J.I. Zucker, ''Processes and the Denotational Semantics of Concurrency,'' *Information and Control* **54**, pp. 70-120 (1982).

[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** (1986).

[Ba2] J.W. de Bakker, ''Comparative Semantics for Flow of Control in Logic Programming without Logic,'' Report CS-R8840, Centrum voor Wiskunde en Informatica, Amsterdam (1988). To appear in Information and Computation.

[JM] N.D. Jones and A. Mycroft, ''Stepwise Development of Operational and Denotational Semantics for Prolog,'' pp. 281-288 in *Proc. Symposium on Logic Programming*, Atlantic City (1984).

[DM] S.K. Debray and P. Mishra, ''Denotational and Operational Semantics for Prolog,'' *Journal of Logic Programming* **5**, pp. 61-91 (1988).

[Bd] M. Badinet, ''Proving Termination Properties of Prolog Programs: A Semantic Approach,'' pp. 336-347 in *Proc. LICS'88*, Edinburgh (1988).

[Vi] E.P. de Vink, ''Comparative Semantics for Prolog with Cut,'' Report IR-166, Vrije Universiteit, Amsterdam (1988). To appear in Science of Computer Programming

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

[Pl] G.D. Plotkin, ''The Category of Complete Partial Orders: a Tool for Making Meanings,'' in *Proc. Summer School on Foundations of Artificial Intelligence and Computer Science*, Pisa (1978).

[BaKo] J.W. de Bakker and J.N. Kok, ''Uniform Abstraction, Atomicity and Contractions in the Comparative Semantics of Concurrent Prolog (Extended Abstract),'' pp. 347-355 in *Proc. International Conference on Future Generation Computer Systems 1988*, Tokyo (1988).