To define the semantics of pairing in Moggi's framework, a product operation $\times : MX \times MY \rightarrow M(X \times Y)$ for the computational strong monad $M$ is needed. After having studied the algebraic properties of the two standard products $\times$ and $\times'$, which correspond to left-to-right and right-to-left evaluation, we have looked for possible alternative products. In the special case, where $M$ is given by the free construction for a decent non-deterministic theory, we found two such alternative products: the strict product $\times$ and the parallel product $\times'$.

Acknowledgments

This paper was written during my one-year visit at Imperial College, London. The visit was made possible by a grant of the Deutsche Forschungsgemeinschaft. I have to thank all the people of the TFM section of Imperial College for their stimulating environment, and for giving me the opportunity to participate in their Section Workshop.

References


On the Equivalence of State-Transition Systems

Michael Huth
mh@doc.ic.ac.uk
Department of Computing, Imperial College
180 Queen's Gate, London SW7 2BZ, United Kingdom

Abstract

We study frameworks for the equivalence of abstract state-transition systems represented as posets. A basic notion of equivalence is proposed. A least fixpoint operator transforms basic equivalences into strong equivalences (=Lagosis Connections) which makes Lagosis Connections into a category. In the absence of divergence, the two notions of equivalence coincide. We generalize these notions by adding a logical level to express divergence more precisely. Then both generalized notions of equivalence coincide even in the presence of divergence.

1 Introduction

We consider posets as abstract state-transition systems. Elements $s$ in a poset $(P, \sqsubseteq)$ model states and $s \sqsubseteq s'$ expresses the intuition that state $s'$ can be reached by performing a finite sequence of state transitions beginning with state $s$. The reasons for including the axiom of antisymmetry will become apparent in this study. Such an approach excludes an immediate analysis of transition graphs; but it does not constitute a conceptual exclusion, for antisymmetry can be restored by rewriting such systems in form of a tree of traces.

What is a reasonable mathematical framework for the equivalence of two such abstract state-transition systems $(P, \sqsubseteq)$ and $(Q, \preceq)$? Assuming that $P$ and $Q$ are intuitively equivalent, we should be able to associate to each state $s$ in $P$ a state $f(s)$ in $Q$ and likewise a state $g(t)$ in $P$ for each state $t \in Q$. So we should impose the existence of a pair of set-theoretic functions $f : P \rightarrow Q$ and $g : Q \rightarrow P$. Further, reachability should be preserved under these state transformations: if $s \sqsubseteq s'$ in $P$, then $f(s) \preceq f(s')$ should follow in $Q$. Therefore, we want $f$ and $g$ to be monotone maps. Finally, we expect the state $g(f(s))$ to be reachable from the original state $s$. Thus, our monotone maps should satisfy the inequalities $id_P \sqsubseteq gf$ and $id_Q \preceq fg$.

Definition 1 Let $P$ and $Q$ be posets. A pair of monotone maps $f : P \rightarrow Q$ and $g : Q \rightarrow P$ is called a poset system [10]; we denote this by $(f, g) : P \rightarrow Q$. We call a poset system a basic equivalence if $id_P \sqsubseteq gf$ and $id_Q \preceq fg$ hold.

In [10], another notion of equivalence had been proposed which is a basic equivalence $(f, g) : P \rightarrow Q$ satisfying the equations $fgf = f$ and $fg = g$, so $f$ and $g$ are additionally quasi-inverses of each other. The functions $gf : P \rightarrow P$ and $fg : Q \rightarrow Q$ are then closure operators with isomorphic image [10]. Such
poset systems are called *Lagois Connections* [10]; they are a generalization of the case where one of the abstract state-transition systems sits as a closure inside the other, i.e., where \( \text{id}_p = gf \) or \( \text{id}_q = fg \). Lagois Connections have a theory similar to the one of *Galois Connections* [4, 9, 10].

**Definition 2** A poset system \((f, g): P \rightarrow Q\) is called a **strong equivalence** if it is a basic equivalence satisfying the equations \( fgf = f \) and \( gfg = g \). □

So the terms **strong equivalence** and *Lagois Connection* have the same meaning. We will address three main issues in this piece of work.

1. Are basic and strong equivalences really equivalences in the mathematical sense?
2. If so, are they really different concepts?
3. How do they match with the most common equivalence results in semantics?

## 2 The Category of Basic Equivalences

There is a natural way of making poset systems into a category [9]; its objects are posets, its morphisms are poset systems \((f, g): P \rightarrow Q\) and \((i, j): Q \rightarrow R\), composition is \((f, g) \circ (i, j) := (if, gj)\) and identities are \((\text{id}_P, \text{id}_P): P \rightarrow P\). Basic equivalences are readily seen to be closed under that composition; so basic equivalences form naturally a category and composition verifies the transitivity of this equivalence notion.

The case of strong equivalences is problematic. Lagois Connections simply are not closed under this composition [10]. But there is an intuitive way of constructing a Lagois Connection out of a basic equivalence \((f, g): P \rightarrow Q\); if we feed back the maps \(f\) and \(g\), we obtain for each \(n \geq 1\) basic equivalences \((f^n, \text{id}, (gf)^n, g^n): P \rightarrow Q\) which form an ascending chain. If our posets have limits of \(\omega\)-chains, then we expect the limit of this chain to be a Lagois Connection. Therefore, we will have to abandon the realm of posets and consider *depocs* in which every directed set has a supremum [5, 7, 12, 15] (of course, \(\omega\)-depocs would suffice).

**Definition 3** Let \(D\) and \(E\) be depocs. A **dcpo system** \((f, g): D \rightarrow E\) is a poset system such that \(f\) and \(g\) are Scott-continuous [5, 7, 12, 15], i.e., they preserve suprema of all directed sets. Define

\[
B(D, E) := \{ (f, g): D \rightarrow E \text{ dcpo system } \mid \text{id}_D \sqsubseteq gf \land \text{id}_E \sqsubseteq fg \} \tag{1}
\]

\[
S(D, E) := \{ (f, g): D \rightarrow E \in B(D, E) \mid fgf = f \land gfg = g \} \tag{2}
\]

and order \(B(D, E)\) and \(S(D, E)\) by

\[
(f, g) \sqsubseteq (i, j) := f \sqsubseteq i \land g \sqsubseteq j. \tag{3}
\]

The letters \(B\) and \(S\) stand for **basic** and **strong** equivalences respectively. If \([D \rightarrow E]\) denotes the function space of all Scott-continuous maps \(f: D \rightarrow E\) in the pointwise order, this is a dcpo where directed suprema of \(\{f_i \mid i \in I\}\) are computed componentwise: \(\{\bigcup_{i \in I} f_i\}(d) = \bigcup_{i \in I} f_i(d)\) [5, 7, 12, 15]. This makes \(B(D, E)\) and \(S(D, E)\) into depocs.

**Proposition 1** Let \(D\) and \(E\) be depocs. Then \(B(D, E)\) is a dcpo and suprema of directed sets are computed componentwise:

\[
\bigcup_{i \in I} (f_i, g_i) = (\bigcup_{i \in I} f_i, \bigcup_{i \in I} g_i). \tag{4}
\]

Moreover, \(S(D, E)\) is a subdcpo of \(B(D, E)\), i.e., the inclusion \(S(D, E) \hookrightarrow B(D, E)\) is Scott-continuous. □

We can recover our poset systems \((f, g): P \rightarrow Q\) from dcpo systems by identifying the posets \(P\) and \(Q\) with the algebraic depocs of their *ideal completions* [4, 7] \(\text{Idl}(P)\) and \(\text{Idl}(Q)\), and by construing \(f\) and \(g\) as Scott-continuous maps which preserve all finite elements. Recall, that a function \(h: D \rightarrow E\) preserves finite elements iff \(h(\text{K}(D)) \subseteq \text{K}(E)\) where \(\text{K}(D)\) denotes the poset of finite, or *compact elements* [4, 5, 7, 17] of a dcpo \(D\).

**Definition 4** For depocs \(D\) and \(E\), define

\[
B^F(D, E) := \{ (f, g) \in B(D, E) \mid f(\text{K}(D)) \subseteq \text{K}(E), g(\text{K}(E)) \subseteq \text{K}(D) \} \tag{5}
\]

\[
S^F(D, E) := B^F(D, E) \cap S(D, E) \tag{6}
\]

in the order induced by \(B(D, E)\). □

The \(F\) in \(B^F\) stands for preserving finite elements. Given algebraic depocs \(D\) and \(E\), dcpo systems in \(B^F(D, E)\), respectively \(S^F(D, E)\), correspond to basic, respectively strong, equivalences between the abstract state-transition systems \(\text{K}(D)\) and \(\text{K}(E)\). Before we make abstract state-transition systems into a category, we should discuss minimal specifications on posets as representations of abstract state-transition systems. First, such a poset \(P\) should satisfy the *descending chain condition* [3]: there do not exist strictly descending chains \(a_0 > a_1 > \cdots\) in \(P\); this means that a state of a computation should be reachable by an initial state (a minimal element of \(P\)). Second, no strictly ascending chain \(b_0 < b_1 < \cdots\) in \(P\) should have an upper bound in \(P\). Otherwise, such an upper bound \(u\) would be a state with an infinite previous history or an infinite cause. We do not, however, exclude the possibility of having infinitely many states that can cause a particular state \(s\): the domain in Figure 1 is a representation of such an abstract state-transition system.

**Definition 5** Let \(\text{ASS}\) be the class of all algebraic depocs \(D\) such that \(\text{K}(D)\) satisfies the descending chain condition and no strictly ascending chain in \(\text{K}(D)\) has an upper bound in \(\text{K}(D)\). Define \(B^F\) to be the four-sorted structure \((\text{ob}(B^F), \text{morph}(B^F), \circ, \text{id})\) such that \(\text{ob}(B^F)\) is the class \(\text{ASS}\), \(\text{morph}(B^F)\) is the class of horn-sets \(B^F(D, E)\) for \(D\) and \(E\) in \(\text{ASS}\), composition is

\[
(f, g) \circ (i, j) := (if, gj). \tag{7}
\]

and identities are

\[
(id_D, id_E): D \rightarrow D. \tag{8}
\]

Define a relation \(\sim\) on \(\text{ASS}\) by

\[
D \sim E \text{ iff } B^F(D, E) \neq \emptyset. \tag{9}
\]

□
This suggests a composition for Lagois Connections. Given dcpo systems $(f, g) \in \mathcal{SF}(D, E)$ and $(i, j) \in \mathcal{SF}(E, F)$, the dcpo system $(f, g) \circ (i, j)$ is a morphism in $\mathcal{BF}(D, F)$, so we could define the strong composition of $(f, g) \in \mathcal{SF}(D, E)$ and $(i, j) \in \mathcal{SF}(E, F)$ to be the least fix point of $\Omega_{(D, F)}$ applied to $(f, g) \circ (i, j)$:

\[
(f, g) \circ (i, j) := \Omega_{(D, F)}((f, g) \circ (i, j)).
\] (15)

Of course, it will be vital to know whether this operation will make $\mathcal{SF}$ into a category. Unfortunately, this is not so in the presence of divergence. For example, consider the dcpo $\omega + 1$ where $K(\omega + 1) = \{0 < 1 < \cdots\}$ is an abstract state-transition system modeling a diverging stream of computation. Define the dcpo systems $(f, f), (g, g) : (\omega + 1) \to (\omega + 1)$ such that $f(n)$, respectively $g(n)$, is the least odd, respectively even, number above $n$. Then $(f, f), (g, g) \in \mathcal{SF}(\omega + 1, \omega + 1)$ is readily seen, but $(f, f) \circ (g, g) = (\lambda n. \omega, \lambda n. \omega)$ is not even in $\mathcal{SF}(\omega + 1, \omega + 1)$ as $\omega$ is not finite in $\omega + 1$.

However, one might ask why $(\lambda n. \omega, \lambda n. \omega)$ does not demonstrate an equivalence on $\omega + 1$? After all, each finite element being mapped to $\omega$ leads only to diverging computations.

3 Handling Divergence with a Logical Level

We therefore propose to add a logical level to dcpo systems $(f, g) \in \mathcal{BD}(D, E)$. If $\mathcal{BD}(D, E)$ denotes the $(f, g) \in \mathcal{BD}(D, E)$ satisfying the logical level still to be defined, we expect for all $D$ and $E$ in ASS, that

- $\mathcal{BD}(D, E)$ is a subset of $\mathcal{BD}(D, E)$,
- $\mathcal{BD}(D, E)$ is closed under $\circ$ in $\mathcal{BD}(D, E)$,
- $\mathcal{BD}(D, E)$ is closed under $\Omega_{(D, E)}$ in $\mathcal{BD}(D, E)$ and
- the relation $\sim$, induced by the sets $\mathcal{BD}(D, E)$ is a computationally meaningful equivalence relation of the class ASS.

Definition 7 For a dcpo $D$, define

\[
D^\omega := D \setminus K(D)
\] (16)

to be the poset of states of divergence of $D$. Given $(f, g) \in \mathcal{BD}(D, E)$, we call $(f, g)$ a demonstration of equivalence iff

\[
1(f^{-1}(E^\omega)) \cap \text{max}(D) \subseteq D^\omega \text{ and } 1(g^{-1}(D^\omega)) \cap \text{max}(E) \subseteq E^\omega,
\] (17)

where $\text{max}(D) := \{d \in D \mid 1(d) = \{d\}\}$. Define

\[
\mathcal{BD}(D, E) := \{(f, g) \in \mathcal{BD}(D, E) \mid (f, g) \text{ is demonstr. of equiv.}\}
\] (19)

in the order induced by $\mathcal{BD}(D, E)$. Define the relation $\sim_\delta$ on ASS by

\[
D \sim_\delta E \text{ iff } \mathcal{BD}(D, E) \neq \emptyset.
\] (21)
The $D$ in $BD$ stands for divergence. The conditions on $f$ say the following (the explanation for $g$ is symmetric and we omit it): if $f$ maps $d \in D$ to a state of divergence, this is fine as long as all maximal states in $D$ above $d$ are states of divergence. For example, $(\lambda n.\omega, \lambda n.\omega) : (\omega + 1) \rightarrow (\omega + 1)$ is such a demonstration of equivalence. However, there is no demonstration of equivalence between $\omega + 1$ and $\{s\}$ as any equivalence has to map $s$ to $\omega \in (\omega + 1)^\infty$, but $s$ is maximal and not a state of divergence. This should suffice to justify the ‘soundness’ of this equivalence notion. We verify that it satisfies our proposed constraints.

**Proposition 2** Let $D$, $E$ and $F$ be elements in $ASS$. Then we have the following.

1. Each $(f, g) \in BD(D, E)$ induces a bijection between $\text{max}(D)$ and $\text{max}(E)$; if $(f, g) \subseteq (i, j)$ in $BD(D, E)$, then $(f, g)$ and $(i, j)$ induce the same bijection.
2. $BD(D, E)$ is an upper set in $BD(D, E)$.
3. $BD(D, E)$ contains $BF(D, E)$ and
4. if $(f, g) \in BD(D, E)$ and $(i, j) \in BD(E, F)$, then $(f, g) \circ (i, j) := (if, gj)$ is in $BD(D, F)$.

Since $BD(D, E)$ is an upper set in $BD(D, E)$ and since $\circ$ is well-defined on $BD$, we can draw several conclusions: $BD$ is a category, $\sim_\circ$ is an equivalence relation, and $\ast$ is a well-defined operation on the hom-sets in $BD$.

**Theorem 3** $BD$ is a self-dual category and $\sim_\circ$ is an equivalence relation on $ASS = \text{ob}(BF)$. Moreover, $BF$ is a full subcategory of $BD$ and $\sim_\circ$ is a subclass of $\sim_\ast$.

**Proof.** The proof that $BD$ is a self-dual category is similar to the corresponding proof for $BF$. Since $BF(D, E) \subseteq BD(D, E)$, the rest is clear.

---

4 **The Category of Lagois Connections**

We have seen that $\ast$ is a well-defined operation on the hom-sets in $BD$. Note that the identities on $BD$ serve also as two-sided identities for $\ast$ applied to strong equivalences as $(id_1, id_\omega)$ is already a strong equivalence for $D$ in $ASS$; but how can we prove that $\ast$ is associative?

The solution follows a general category-theoretical pattern which we have so far not been able to match with standard concepts in category theory [2, 8].

**Definition 8** Let $C$ be a category and $\Omega$ a class of set-theoretic functions

$$\Omega(A, B) : C(A, B) \rightarrow C(A, B)$$

for all objects $A$ and $B$ in $C$. We call $\Omega$ an associative structure over $C$ iff for all morphisms $f \in C(B, C)$ and $g \in C(A, B)$, we have

$$\Omega(A, C)(f \circ (\Omega(A, B)g)) = \Omega(A, C)(f \circ g)$$

(23)

and

$$\Omega(A, C)((\Omega(B, C)f) \circ g).$$

(24)

Define

$$C[\Omega](A, B) := \text{im}(\Omega(A, B)) = \{\Omega(A, B)f \mid f \in C(A, B)\}$$

(25)

$$id^\Omega_A := \Omega(A, A)id_A$$

(26)

and

$$f \ast g := \Omega(A, C)(f \circ g).$$

(27)

The four-sorted structure $(\text{ob}(C), C[\Omega], \ast, id^\Omega)$ is a category and $\Omega$ induces a functor $\Omega : C \rightarrow C[\Omega]$.

**Theorem 4** Let $C$ be a category and $\Omega$ an associative structure over $C$. Then we have the following:

1. For all objects $A$ and $B$ in $C$, the function $\Omega(A, B)$ is a retraction on $C(A, B)$,
2. the four-sorted structure $(\text{ob}(C), C[\Omega], \ast, id^\Omega)$ is a category and
3. if $\Omega$ also denotes the function which leaves objects of $C$ fixed and maps morphisms $f \in C(A, B)$ to $\Omega(A, B)f$, then $\Omega$ is a functor

$$\Omega : C \rightarrow C[\Omega].$$

(28)

We will now show that the class of Scott-continuous closure operators of Theorem 2 is an associative structure over the category $BD$. This will make Lagos Connections into a category. First, we will reduce the complexity of such a proof; note that the operator $\Omega$ occurs nested in the equations (23)–(24). We can use the concrete poset structure of the category $BD$ to reduce these equations to equivalent inequalities in which $\Omega$ only appears at one level.

**Proposition 3** Let $C$ be a category such that all hom-sets $C(A, B)$ are posets and composition is monotone. If $\Omega$ is a class of closure operators $\Omega(A, B)$ on $C(A, B)$ for all objects $A$ and $B$ in $C$, then $\Omega$ is an associative structure over $C$ if for all morphisms $f \in C(B, C)$ and $g \in C(A, B)$, we have

$$f \circ (\Omega(A, B)g) \subseteq \Omega(A, C)(f \circ g)$$

(29)

and

$$\Omega(\Omega(B, C)f) \circ g \subseteq \Omega(A, C)(f \circ g).$$

(30)

In that case, $\ast$ is monotone.

**Theorem 5** The class of Scott-continuous closure operators $\Omega$ of Theorem 2 is an associative structure over the category $BD$. In particular,

$$SD := BD[\Omega]$$

(31)

is a category, the category of Lagos Connections. The composition in $SD$ is Scott-continuous and is given by the feedback formula

$$(f, g) \ast (i, j) = \{i | (gf)^n(i) = (ij)^n(gj)\}. \quad (32)$$

**Proof.** The assumptions of Proposition 3 apply so it suffices to show the inequalities (29)–(30). For $(f, g) \in BD(D, E)$ and $(i, j) \in BD(E, F)$, we compute the first coordinates of expressions occurring in these inequalities. The proof for the second coordinates is similar and we omit it. The first coordinate of $\Omega(D, F)((f, g) \circ (i, j))$ equals $\Omega_{\leq x}(i)$ and the first coordinate of $(f, g) \circ (\Omega(E, F)(i, j))$ equals $\Omega_{\leq x}(i)$ and the first coordinate of $(\Omega(D, F)(i, j) \circ (i, j))$ equals $\Omega_{\leq x}(i)$ and the first coordinate of $(ifg)^n(i)$, so we are done if $(ifg)^n(i, j)$.

$$f \ast g := \Omega(A, C)(f \circ g).$$

(27)

for all $n \geq 1$; this is an easy induction.

$$\Omega : C \rightarrow C[\Omega].$$

(28)
One could now define an equivalence relation \( \approx \) such that \( D \approx E \) in ASS iff \( SD(D,E) \neq \emptyset \). But since \( \Omega(D,E)(BD(D,E)) = SD(D,E) \), this relation equals \( \sim \). Thus, these relations render the same concept of equivalence even in the presence of divergence. What can we say in the absence of divergence? If TERM denotes the class of all \( D \) in ASS such that \( K(D) = D \), then we have the set equalities \( BF(D,E) = BD(D,E) = B(D,E) \) as there are no states of divergence for \( D \) and \( E \) in TERM.

**Theorem 6** On the class TERM, the relations \( \sim \) and \( \approx \) are equal; where we define

\[
D \approx E \text{ iff } SF(D,E) \neq \emptyset. \tag{33}
\]

In particular, \( \approx \) is an equivalence relation on TERM.

One might ask whether \( \approx \) is also transitive on ASS, or whether its transitive closure equals \( \sim \) or \( \approx \); we have not investigated this any further.

The feed-back formula intrinsically contains cases of nilpotency which had been noted in [10] as instances of a well-defined composition for LaRoi's Connections.

**Proposition 4** For \( D \) and \( E \) in ASS and \((f,g) \in BD(D,E)\), we have

1. \((fg,fg) \in BD(D,E)\), and \((gf,gf) \in BD(D,D)\),
2. if \( f g f = f \), then
   \[
   \Omega(D,E)(f,g) = (f,gf),
   \tag{34}
   \]
3. if \( g f g = g \), then
   \[
   \Omega(D,E)(f,g) = (gf,g),
   \tag{35}
   \]
4. if \( f g \) or \( g f \) is idempotent, then
   \[
   \Omega(D,E)(f,g) = (gf,gf).
   \tag{36}
   \]
\[
\Omega(E,E)(fg,fg) = (fg,fg) \quad \text{and} \quad \Omega(D,D)(gf,gf) = (gf,gf).
\tag{37}
\]

Let us point out the conceptual gain of these results. By acknowledging the logical level as a sound criterion of equivalence, we can prove two objects \( D \) and \( E \) in ASS to be equivalent by specifying a basic equivalence \((f,g) : D \rightarrow E \) in \( BD(D,E) \). If we prefer to work with a LaRoi's connection, we can simply take \((f,g)\) and compute the least fix point \( \Omega(D,E)(f,g) \) which will be an element of \( SD(D,E) \).

There is a larger framework than the categories \( BD \) and \( SD \) which does not model equivalences of abstract state-transition systems, but which does have a mathematical interest in its own right. If \( F \) denotes the four-sorted structure which has all dcpos as objects, dcpo systems as morphisms, \( \sigma \) as composition and pairs \((id_{\alpha}, id_{\beta})\) as morphisms, then \( B \) is a category which contains \( BD \) as a non-full subcategory. Likewise, if we consider

\[
S := B[0]
\tag{39}
\]

then we obtain a corresponding version of Theorem 5 for \( B \) and \( S \).

## 5 Basic Equivalences in Semantics

We want to relate the concepts and results of the preceding sections to familiar situations in formal semantics of programming languages. As a first example, let us compare operational semantics for a simple deterministic, imperative WHILE-language [5, 12, 15]. The concrete choice of such a language is irrelevant for the purpose of this discussion. We only need to know that states for such a language come in two flavors. A state is either a pair \((S,s)\) where \( S \) is a statement in the language and \( s \) denotes an environment which binds free variables in \( S \), or it is a terminal state \( s' \). Intuitively, a state \( s' \) models the termination of a program's response to an initial environment. Assuming that our programming language does not produce any stuck configurations [12], a state \((S,s)\) leads to a successor state which is either terminal, i.e., of the from \( s' \), or again of the form \((S',s')\).

There are two prominent operational semantics for such WHILE-languages, called structural operational semantics and natural semantics [12]. The structural operational semantics is defined by a relation \( \Rightarrow \) which specifies what the successor states are; then \( \Rightarrow^* \) is the transitive closure of \( \Rightarrow \). This semantics models a single-step computation. The natural semantics models a big-step computation. We write \((S,s) \rightarrow s'\) if the computation starting at \((S,s)\) ends in state \( s' \). Otherwise, the relation \( \rightarrow \) is undefined. Let \( NS \) be the preorder obtained by forming the reflexive closure of \( \rightarrow \) on the set of states of our WHILE-language, and let \( SOS \) be the preorder obtained by forming the reflexive closure of \( \Rightarrow^* \) on the same set. We have functions \( f : NS \rightarrow SOS \) and \( g : SOS \rightarrow NS \) defined by

\[
f(S,s) := (S,s) \tag{40},
\]
\[
f(s') := s' \tag{41},
\]
\[
g(s') := s' \text{ and} \tag{42},
\]
\[
g((S,s)) := \text{ if } (S,s) \Rightarrow^* s' \text{ then } s' \text{ else } (S,s). \tag{43}
\]

We readily check that \( f \) is monotone; this means that \((S,s) \rightarrow s'\) implies \((S,s) \Rightarrow^* s'\), so the big-step deduction is sound with respect to the single-step deduction. We also have \( id_{NS} \subseteq gf \) because \((S,s) \Rightarrow^* s'\) guarantees \((S,s) \rightarrow s'\). Further, we obtain \( id_{SOS} \subseteq fg \) and the equations \( fgf = f \) and \( gfg = g \). So it looks as if

\[
(f,g) : NS \rightarrow SOS \tag{44}
\]

is a LaRoi's Connection between the preorders \( NS \) and \( SOS \). Yet, we did not prove the monotonicity of \( g \); but \( g \) simply is not monotone. Consider a transition \((S,s) \Rightarrow^* (S',s')\) such that the computation according to \( \Rightarrow^* \) is diverging after \((S,s)\), and therefore after \((S',s')\) as well. By definition,
\[
g((S,s)) = (S,s) \text{ and } g((S',s')) = (S',s'),
\]
but \(-\) is a discrete order on the set of states \((S,s)\) which give rise to diverging behavior. So our example is really a good non-example. The fact that \( g \) is not monotone captures the essential difference between these two operational semantics: the one-step ordering on diverging streams has no equivalent in the big-step computation. It is worth pointing out that the mere existence of \( f \) and \( g \) suffice to prove the usual semantic correspondence theorems [12]; the non-monotone behavior of \( g \) does not affect this.
It is possible to state another big-step operational semantics such that we do have a Lagois Connection between it and the structural operational semantics. For that, let D be the ideal completion of SOS; so we just add one limit point for each diverging stream \((S_0, s_0) \Rightarrow (S_1, s_1) \Rightarrow \cdots\). Since our language is assumed to be deterministic, the relation \(\Rightarrow\) can be viewed as a Scott-continuous function \(h: D \rightarrow D\) such that

\[
h(S, s) \equiv s' \iff (S, s) \Rightarrow s',
\]

\[
h(S, s) \equiv (S', s') \iff (S, s) \Rightarrow (S', s') \text{ and}
\]

\[
h(s') \equiv s'.
\]

Since \(iD_0 \subseteq h\) is obvious, we conclude \((h, h) \in BF(D, D)\). Therefore, we can define

\[
\langle \text{bigstep, bigstep} \rangle = \Omega_p(D, D)\langle h, h \rangle
\]

which is an element in \(SD(D, D)\). Moreover, \(\text{bigstep}\) is a Scott-continuous closure operator on \(D\). Its image consists of all elements of \(D^\omega\) and all terminal states \(s'\). The \(n\)-th approximation \((h_n, h_n)\) of the fix point \((\text{bigstep, bigstep})\) computes \(2n + 1\) steps from a given state \((S, s)\) as

\[
\langle h_n, h_n \rangle = ((hh)^n h, (hh)^n h) = (h^{2n+1}, h^{2n+1}).
\]

It would be interesting to relate operational semantics and abstract machines [12] using the concepts of basic and strong equivalences. We will not do this here for lack of space. However, let us remark that certain configurations on abstract machines might not correspond to anything meaningful in a given operational semantics. Thus, one should study the mathematical framework proposed in this paper in a setting of partial equivalences. We have not yet investigated this any further.

Given an abstract machine for our WHILE-language, one usually proves that the semantic function induced by this machine is equivalent to the one induced by the natural semantics. However, one can also prove this using the structural operational semantics [12]. This is being done by introducing a \(\text{bisimulation}\) relation [12]. It seems as if the specification of such a relation is nothing else than the definition of a Lagois Connection between the respective preorders of states, for Lagois Connections \((f, g): P \rightarrow Q\) can be characterized as certain equivalence relations on \(P\) and \(Q\) [10]. This apparent analogy needs further study.

A simple example of a strong equivalence between an abstract machine and an operational semantics can be found in [10]. It relates an operational semantics of \(\text{marked infix arithmetic expressions}\) to a stack machine for evaluating \(\text{postfix arithmetic expressions}\).

In [6], an abstract semantics for a higher-order functional language with logic variables has been given by solving \(\text{simultaneous fix-point equations of closure operators over Scott-domains}\). Since constraints are modeled by closure operators [6], one wants to associate a closure operator \(h\) with two constraints modeled by \(f\) and \(g\). It turns out that this can be viewed as

\[
\langle h, h \rangle = \langle f, f \rangle \ast \langle g, g \rangle.
\]

There is an underlying symmetry for the composition \(\ast\) if all participants are closure operators represented as in equation (50). This reflects the permutation symmetry of a finite set of constraints as the composition \(\ast\) is then \(\text{commutative}\). Our mathematical framework could serve as a 'clean' approach to this kind of constraint semantics.

We saw above that the existence of a Lagois Connection between two operational semantics, where one map was only partially monotone, brought about a correspondence theorem between them: the semantic functions induced by them are extensionally equal. The existence of a basic equivalence is actually stronger than the extensional equality of semantic functions between the respective state-transition systems. For example, let \(D\) denote the dcpo in ASS which has one initial state \(0\) and then two possible successor states \(a_0\) and \(b_0\) which initiate diverging streams of computation \(a_0 < a_1 < \cdots\) and \(b_0 < b_1 < \cdots\). The semantic function of \(D\) equals the one of \(\omega + 1\), namely \(\lambda s.1\) as no computation terminates. A basic equivalence \((f, g): D \rightarrow (\omega + 1)\) would induce bijections between \(\max(D)\) and \(\max(\omega + 1)\), but the first set has two elements and the second one is a singleton. Therefore, basic equivalences not only control terminating behavior, they also guarantee that the pattern of diverging streams coincide.

Also, having a bijection between \(\max(D)\) and \(\max(E)\) is not sufficient to conclude \(D \sim E\). As an example, take the domains \(D_0\) and \(D_1\) in Figure 2 which have each two maximal elements. It is readily seen that there is no basic equivalence \((f, g): D_0 \rightarrow D_1\).

![Figure 2: dcpo\(D_0\) and \(D_1\) in ASS with \(\max(D_0) \equiv \max(D_1)\) such that there is no basic equivalence \((f, g): D_0 \rightarrow D_1\)]

\[\text{Figure 2: dcpo } D_0 \text{ and } D_1 \text{ in ASS with } \max(D_0) \equiv \max(D_1) \text{ such that there is no basic equivalence } (f, g): D_0 \rightarrow D_1.\]

6 Conclusions

We established a category \(BD\) with abstract state-transition systems as objects and basic equivalences as morphisms. We introduced the concept of an associative structure on a category which lead to the category \(SD\) with abstract state-transition systems as objects and Lagois Connections as morphisms. The categorical structures gave us two equivalence relations; since basic equivalences are transformed into Lagois Connections by a least fix-point operator, these two equivalence relations coincide. We discussed structural operational semantics and natural semantics in light of these results. The supremum of closure operators is the composition in \(SD\). This gives an interactive account of a semantics of constraint programming.
Acknowledgements

Thanks to Austin Melton for sending me a copy of [10] so fast. Samson Abramsky made most valuable comments throughout the progress of this work. Olivier Danvy was a critical and helpful reader of this article. Radha Jagadeesan pointed out to me the relevance of the composition * to his work in [6]. Geoffrey Burn and Abbas Edalat helped shaping this paper as reviewers.

References


Towards a Modal Logic of Durative Actions

Stuart Kent
sjk@doc.ic.ac.uk
Department of Computing, Imperial College
180 Queen's Gate, London SW7 2BZ, United Kingdom

Abstract

This paper proposes an extension of modal action logics, which typically make the assumption that an action is atomic, to include durative actions. These logics have been developed to support the formal specification of information systems: we argue, with particular reference to object oriented systems, that assuming atomicity is too restrictive to express many kinds of temporal constraint. In consequence, we propose that actions be regarded as durative, and encode this by assuming that an action occurs over a sequence of atomic transitions, or interval, rather than a single transition. With this as a pre-requisite, the paper continues to redefine and extend operators of atomic action logics to fit the durative case.

1 Introduction

This paper describes work whose aim is to support the formal specification, characterisation and development of object oriented systems. Specifically, we propose an extension to action logics [16, 18, 7, 15] which have already been used with some success in providing axiomatic semantics to object oriented specification languages [8, 9] and thereby providing a basis for reasoning about object oriented systems and their development. The core of our proposal is a reworking of the semantics of such logics to admit durative actions.

An action logic distinguishes between terms denoting actions performed in the system and terms denoting values. The value-denoting terms are used to represent the state of the "abstract machine" being specified. Actions identify transitions to change that state. Typically, the semantics of these logics force actions to be atomic; they are regarded as occurring over a single atomic transition and, with regard to concurrency, it is only possible to express restrictions on their synchronisation. However, when constructing an object oriented model of a system one usually makes the assumption that methods have duration, and, in general, may operate concurrently not just synchronously – i.e. one method may start whilst another is in progress, and only parts of a method need to synchronise where conflict avoidance is required. This is reflected in the recent development of OO programming languages, such as DRAGON [4] and POOL [3], in which concurrent method execution is the norm and language

1 Perhaps an exception to this is [17], which admits compound durative actions (e.g. actions constructed using a sequential combinator). However, primitive durative actions (e.g. actions whose structure is not determined) are not allowed.