Semantic Models for a Language with Timed Atomic Actions

Franck van Breugel

Department of Mathematics and Computer Science
Vrije Universiteit
De Boelelaan 1081a
1081 HV Amsterdam
email: franck@cs.vu.nl

Abstract

An operational and a denotational semantic model for a language incorporating time related aspects, viz. timed atomic actions and integration, is presented. With timed atomic actions we mean atomic actions each provided with a time stamp, which records the time at which the atomic action should be executed. Integration of a statement over some non-empty subset of the time domain gives rise to the execution of the statement with some non-deterministically chosen value of this subset passed to that statement. Both models are built on complete metric spaces. An equivalence result of the operational and the denotational semantic model concludes this paper.

1 Introduction

In this paper the question whether programming notions expressing time related behaviour can be described by models built on structures from metric topology in the style of De Bakker et al. is studied. In this style, programming notions related to concurrency [BZ82, BBKM84, BKMOZ86, KR88], object-oriented programming [ABKR86, AB88] and logic programming [B88, BK88] have already been described.

An operational and a denotational semantic model for a simple language incorporating time related aspects is given. Also an equivalence result for these two semantic models is presented.

The language studied is a uniform language built from timed atomic actions, sequential composition, non-deterministic choice, parallel composition, integration and recursion. The time related aspects of this language are timed actions and integration. With timed atomic actions we mean atomic actions each provided with a time stamp. This absolute time stamp denotes when the atomic action should be executed. Whenever an atomic action cannot be executed at the time recorded in its time stamp a so-called failure occurs and the execution of the program stops. Integration of a statement over some time set, i.e. a non-empty finite subset of the time domain, gives rise to the execution of the statement with some non-deterministically chosen value of the time set passed to that statement.

We assume that the execution of timed atomic actions and operators takes no time. We also assume that two successive atomic actions cannot be executed at the same time. These assumptions are very useful as is argued in a paper by Berry and Cosserat [BC85].

The operational model is based on Plotkin's transition systems as introduced by Hennessy and Plotkin in [HP79]. This operational semantic model is a modified version of the model presented by Baeten and Bergstra in [BB89].

The denotational model is defined compositionally using fixed points to deal with recursion. Such fixed points exist on the basis of Banach's theorem for contracting functions on complete metric spaces [KR88, BM88]. Most definitions and lemmas in this paper rely on this theorem.

Related work can be found in [RR86, RR87, R88] of Reed and Roscoe and [LZ88] of Lee and Zwarico. In [RR86, RR87, R88] denotational semantic models based on metric spaces are presented for the language TCSP, as described by Davies and Schneider in [DS89]. All models incorporate timed traces. A timed trace is a finite trace of timed events which records the history of the execution of the program. Reed and Roscoe introduce a system delay constant such that the execution of a program amounts to performing only finitely many actions in a bounded period of time. This system delay constant is also used to introduce some guardedness with respect to procedure bodies. Although the language TCSP differs from the language studied in this paper, the denotational semantic models for TCSP show some resemblance to our denotational model. In [LZ88], Lee and Zwarico present a model which is also based on timed traces. They use a temporal extension of acceptance trees to model the execution of a program

In the second section of this paper the language definition is given. The operational and denotational semantics are defined in respectively section 3 and 4. Section 5 contains the equivalence result. The final section consists of some concluding remarks. Some mathematical preliminaries concerning metric spaces and labelled transition systems can be found in the appendix.

2 Language definition

In this section we introduce the syntax of the language $Prog$, which is studied in this paper. This language incorporates some time related aspects. We build the syntax starting from the following sets.

- A set $Atom$ of atomic actions, with typical element a, which can be infinite.
- The set $\mathbb{R}_>$, the set positive real numbers, with typical element r, which is our time domain.
- A collection of time sets, which is represented by $\mathcal{P}_{nf}\ (\mathbb{R}_>)$, the set of non-empty finite subsets of $\mathbb{R}_>$, with typical element T.
- A set $Tvar$ of time variables, with typical element t.
- A set $Pvar$ of procedure variables, with typical element x.

Each procedure variable x has a certain arity n, with $n \geq 0$, and is labelled with n distinct time variables: $x\ (t_1, \ldots, t_n)$. We can view these time variables as formal parameters.

We introduce the syntactic class Exp of expressions, with typical element e. For simplicity, we assume the syntax of Exp built from elements of $Tvar$ and some arithmetic operators. We postulate that no complications arise in the evaluation of expressions. The evaluation of an expression is given by the evaluation function $\mathcal{E}$. We apply this evaluation function only to expressions without free variables. In that case, the function $\mathcal{E}$ delivers an element of $\mathbb{R}$. For example we have that $\mathcal{E}\ (3 + 4) = 7$.

Now we can give the class of statements.

Definition 2.1
The class $Stat$ of statements, with typical element s, is given by
$s ::= (a, e) \mid x\ (e_1, \ldots, e_n) \mid s_1\ ;\ s_2 \mid s_1 \cup s_2 \mid s_1 \parallel s_2 \mid \int_{t \in T} s$
End 2.1

A statement s is of one of the six following forms:

- (a, e)
  A timed atomic action: the atomic action a has to be executed at time $\mathcal{E}\ (e)$. However, the execution of an atomic action at a non-positive time always fails.
- $x\ (e_1, \ldots, e_n)$
  A procedure call: execution of the corresponding body of the procedure with the expressions $e_1, \ldots, e_n$ passed to the procedure body.
- $s_1\ ;\ s_2$
  Sequential composition of the statements s_1 and s_2.
- $s_1 \cup s_2$
  Nondeterministic choice of the statements s_1 and s_2.
- $s_1 \parallel s_2$
  Parallel composition, or merge, of the statements s_1 and s_2: the arbitrary interleaving of the atomic actions of both statements.
- $\int_{t \in T} s$
  Integration: execution of statement s with an arbitrary element of T passed to time variable t in s.

To guarantee that we can always take an arbitrary element of a time set, we have to restrict time sets to non-empty subsets of $\mathbb{R}_>$. The restriction of time sets to finite subsets of $\mathbb{R}_>$ will be crucial in the proof of property 3.4.

The execution of the integration $\int_{t\in\{1,2,3\}} (a,\ t)$ corresponds to the execution of the statement $(a,\ t)$ with a non-deterministically chosen value of the time set {1,2,3} passed to the time variable t in that statement, which gives us the execution of $(a,\ 1)$, $(a,\ 2)$ or $(a,\ 3)$. Note that the integration is a generalisation, due to the value passing, of the non-deterministic choice.

We assume that the execution of atomic actions and operators takes no time. We further assume that two successive atomic actions cannot be executed at the same time. These assumptions are very useful as is argued in a paper by Berry and Cosserat [BC85].

Next we introduce the class of guarded statements, which will be used to administrate procedure bodies.

Definition 2.2
The class $Gstat$ of guarded statements, with typical element g, is given by
$g ::= (a,\ e) \mid g\ ;\ s \mid g_1 \cup g_2 \mid g_1 \parallel g_2 \mid \int_{t\in T} g$
End 2.2

Before we give the definition of the class of declarations, we introduce the notion of free variables.

Definition 2.3
The mapping $fv : Stat \rightarrow \mathcal{P}\ (Tvar)$ is given by
$fv\ ((a,\ e)) = fv\ (e)$
$fv\ (x\ (e_1,\ \ldots,\ e_n)) = fv\ (e_1) \cup \ldots \cup fv\ (e_n)$
$fv\ (s_1\ op\ s_2) = fv\ (s_1) \cup fv\ (s_2)$ $\qquad op \in \{;,\cup,\parallel\}$
$fv\ (\int_{t\in T}\ s) = fv\ (s) \setminus \{t\}$
End 2.3

Definition 2.4
The class $Decl$ of declarations, with typical element d, consists of m-tuples
$d \equiv x_1\ (t_1,\ \ldots,\ t_{n_1}) \Leftarrow g_1,\ \ldots,\ x_m\ (t_1,\ \ldots,\ t_{n_m}) \Leftarrow g_m$,
where x_i are distinct procedure variables of arity n_i, $t_1, \ldots t_{n_i}$ are distinct elements of $Tvar$, $g_i \in Gstat$ and $fv\ (g_i) \subseteq \{t_1,\ \ldots,\ t_{n_i}\}$.
End 2.4

All procedure bodies g_i in a declaration d are restricted to guarded statements. This requirement corresponds to the usual Greibach condition in formal language theory. There are possibilities to eliminate this restriction as is illustrated in [RR86, RR87] and [R88] by Reed and Roscoe. However, in order to permit unguarded statements in declarations, we expect that we have to determine some fixed delay constant for some operators. The restriction $fv\ (g_i) \subseteq \{t_1,\ \ldots,\ t_{n_i}\}$ states that there should not be any global time variables.

The execution of the procedure call $x\ (1)$, where $x\ (t)$ is declared as $(a,\ t)\ ;\ x\ (\mathrm{t} + 1)$, corresponds to the execution of the procedure body $(a,\ t)\ ;\ x\ (\mathrm{t} + 1)$ with the expression 1 passed to this procedure body, i.e. $(a,\ 1)\ ;\ x\ (1 + 1)$.

Next we give the definition of the class of programs.

Definition 2.5
The class $Prog$ of programs, with typical element p, consists of pairs
$p \equiv d \mid s$,
where $d \in Decl$, $s \in Stat$ such that each procedure variable x occurring in s or d is declared in d and s contains no free variables.
End 2.5

It is possible to execute an infinite number of atomic actions in a finite amount of time. For example, the program $x\ (t) \Leftarrow (a,\ t)\ ;\ x\ (t + \frac{t}{2}) \mid x\ (\frac{1}{2})$ performs an infinite number of a's between 0 and 1.

3 Operational semantics

In this section we present an operational semantics for our language. This operational semantic model is a modified version of the model defined by Baeten and Bergstra in [BB89]. The operational semantics of a program describes the behaviour of an abstract machine running that program. The execution of a program on an abstract machine is characterised by strings of actions. Which actions and in which order the actions are performed by the abstract machine is described by means of a labelled transition system.

Before introducing the labelled transition system, we first take a look at the phenomenon of termination of a program. We can distinguish two reasons for termination: (i) termination if the program has successfully executed all its atomic actions, which we call successful termination, and (ii) termination if an atomic action of the program cannot be executed at the time it should be executed, which we call failure. For example, $(a,\ 4)\ ;\ (b,\ 3)$ will fail immediately after the execution of the atomic action a. Failure is modelled by a special symbol δ with $\delta \notin Atom$. We introduce two new statements linked to these notions of termination. The empty statement E is associated with successful termination as introduced by Apt in [A81] and the failure statement Δ is associated with failure. Δ and E have a different nature as will become clear in the equivalences given below and in the equivalence proof presented in section 5. The set of statements $Stat$ is extended to $Stat_{E\Delta}$.

$Stat_{E\Delta} = Stat \cup \{E, \Delta\}$
with typical element $\bar{s}$.

It will be convenient to allow expressions of the form $\bar{s}\ ;\ \bar{s}'$ and $\bar{s} \parallel \bar{s}'$ for $\bar{s},\ \bar{s}' \in Stat_{E\Delta}$. This will reduce the number of rules of the labelled transition system. We define the following reasonable equivalences on these expressions for $\bar{s} \in Stat_{E\Delta}$ and $op \in \{;, \parallel\}$.

$\bar{s}\ op\ E = \bar{s}$
$E\ op\ \bar{s} = \bar{s}$
$\bar{s}\ op\ \Delta = \Delta$
$\Delta\ op\ \bar{s} = \Delta$

Now we can introduce the labelled transition system. The labelled transition system is associated with a declaration d as will become clear in the definition of the rules. The set of configurations consists of the set

$Conf_{E\Delta} = Stat_{E\Delta} \times I\!R_{\geq}$
with typical typical element $\bar{C}$.

In a configuration $[\bar{s},\ r]$, r denotes the global time at which the atomic action preceding the execution of statement $\bar{s}$ was executed. Global time is administrated to fulfill the restriction that two successive atomic actions cannot be executed at the same time. We also introduce the set

$Conf = Stat \times I\!R_{\geq}$
with typical element C.

Before we give the set of labels we define the set of timed actions and the set of timed failures.

The set TA of timed actions is defined by
$TA = Atom \times I\!R_{>}$
with typical element α.
The set TF of timed failures is defined by
$TF = \{\delta\} \times I\!R_{\geq}$.

The set of labels consists of the set

$Label = TA \cup TF$

with typical element l.

The label $(a,\ r)$ denotes that the atomic action a is executed at time r. The label $(\delta,\ r)$ denotes failure at time r.

Intuitively, a rule $[\bar{s},\ r] -l\rightarrow_d [\bar{s}',\ r']$ tells us that the execution of statement $\bar{s}$, following an atomic action executed at time r, consists of action l at time r' followed by the execution of statement $\bar{s}'$. Now we give the set of rules associated with the declaration d.

$$[(a,\ e),\ r] -(a,\ \epsilon)\rightarrow_d [E,\ \epsilon] \qquad \mathcal{E}\ (e) = \epsilon \text{ and } \epsilon > r$$

$$[(a,\ e),\ r] -(\delta,\ r)\rightarrow_d [\Delta,\ r] \qquad \mathcal{E}\ (e) = \epsilon \text{ and } \epsilon \leq r$$

$$\frac{[g\ [e_1/t_1,\ \ldots,\ e_n/t_n],\ r] -l\rightarrow_d \bar{C}}{[x\ (e_1,\ \ldots,\ e_n),\ r] -l\rightarrow_d \bar{C}} \qquad x\ (t_1,\ \ldots,\ t_n) \Leftarrow g \in d$$

$$\frac{[s,\ r] -l\rightarrow_d [\bar{s},\ r']}{[s\ ;\ s',\ r] -l\rightarrow_d [\bar{s}\ ;\ s',\ r']}$$

$$\frac{[s,\ r] -l\rightarrow_d \bar{C}}{\begin{array}{l}[s \cup s',\ r] -l\rightarrow_d \bar{C} \\ [s' \cup s,\ r] -l\rightarrow_d \bar{C}\end{array}}$$

$$\frac{[s,\ r] -l\rightarrow_d [\bar{s},\ r']}{\begin{array}{l}[s \parallel s',\ r] -l\rightarrow_d [\bar{s} \parallel s',\ r'] \\ [s' \parallel s,\ r] -l\rightarrow_d [s' \parallel \bar{s},\ r']\end{array}}$$

$$\frac{[s[r'/t],\ r] -l\rightarrow_d \bar{C}}{[\int_{t\in T} s,\ r] -l\rightarrow_d \bar{C}} \qquad r' \in T$$

Consider the axioms for the configuration $[(a,\ e),\ r]$, where $\epsilon = \mathcal{E}\ (e)$. At time r the atomic action preceding the execution of the statement $(a,\ e)$ was executed. The time ϵ denotes at which time atomic action a should be executed. Since two successive atomic actions cannot be executed at the same time, the atomic action a can only be executed at time ϵ and terminate successfully at time ϵ, which is denoted by $[E,\ \epsilon]$, if $\epsilon > r$. If $\epsilon \leq r$, the atomic action a should be executed at or before the time at which the atomic action preceding the execution of the statement $(a,\ e)$ was executed. In this case, the execution of the atomic action a fails at time r, denoted by $[\Delta,\ r]$. We will only use the axioms for the configuration $[(a,\ e),\ r]$ in the case that e does not contain any free variables, because we impose that the statement part of a program does not contain any free variables.

The rule for a procedure call indicates body replacement and value passing of the expressions to the time variables. Parallel composition is modelled by arbitrary interleaving of the atomic actions of both statements. The rule for integration states that some element r' from the time set T is passed to the time variable t in statement s.

Using the above rules, we can derive $[(a,\ 3)\ ;\ (b,\ 4),\ 0] -(a,\ 3)\rightarrow_d [(b,\ 4),\ 3] -(b,\ 4)\rightarrow_d [E,\ 4]$ and $[(b,\ 4)\ ;\ (a,\ 3),\ 0] -(b,\ 4)\rightarrow_d [(a,\ 3),\ 4] -(\delta,\ 4)\rightarrow_d [\Delta,\ 4]$.

Now we define the operational semantics for configurations. The strings of actions obtained from the labelled transition system are recorded by the mapping $\mathcal{O}'_d$. These strings of actions are collected in $\mathcal{P}_{nc}\ (TFS')$, the set of non-empty closed subsets of

$TFS' = Label^+ \cup Label^\omega$
with typical element σ.

As we will see in property 3.3, for each configuration the operational semantics is a non-empty set. We restrict ourselves to closed sets in order to obtain a complete metric space. At the end of this section the semantic domain is restricted.

Definition 3.1
The mapping $\mathcal{O}'_d : Conf \to \mathcal{P}_{nc}(TFS')$ is given by
$\sigma \in \mathcal{O}'_d(C)$ if and only if one of the following three conditions is satisfied

1. $\exists\, n \in I\!N\ \exists\, C_1, \ldots, C_{n-1} \in Conf\ \exists\, l_1, \ldots, l_n \in Label\ \exists\, r \in I\!R_{\geq}$
   $C -l_1\to_d C_1 -l_2\to_d \ldots -l_{n-1}\to_d C_{n-1} -l_n\to_d [E, r] \wedge$
   $\sigma = l_1\ l_2\ \ldots l_{n-1}\ l_n$
2. $\exists\, n \in I\!N\ \exists\, C_1, \ldots, C_{n-1} \in Conf\ \exists\, l_1, \ldots, l_n \in Label\ \exists\, r \in I\!R_{\geq}$
   $C -l_1\to_d C_1 -l_2\to_d \ldots -l_{n-1}\to_d C_{n-1} -l_n\to_d [\Delta, r] \wedge$
   $\sigma = l_1\ l_2\ \ldots l_{n-1}\ l_n$
3. $\exists\, C_1, \ldots \in Conf\ \exists\, l_1, \ldots \in Label$
   $C -l_1\to_d C_1 -l_2\to_d \ldots -l_{n-1}\to_d C_{n-1} -l_n\to_d \ldots \wedge$
   $\sigma = l_1\ l_2\ \ldots l_{n-1}\ l_n\ \ldots$

End 3.1

To prove that $\mathcal{O}'_d$ is well-defined, we need two properties of the labelled transition system. We prove these properties using induction on the complexity of statements. We define a complexity function on statements associated with a declaration d.

Definition 3.2
The mapping $cf_d : Stat \to I\!N$ is given by
$cf_d((a, e)) = 1$
$cf_d(x\ (e_1, \ldots, e_n)) = cf_d(g) + 1 \qquad x\ (t_1, \ldots, t_n) \Leftarrow g \in d$
$cf_d(s_1\ ;\ s_2) = cf_d(s_1) + 1$
$cf_d(s_1 \cup s_2) = cf_d(s_1) + cf_d(s_2)$
$cf_d(s_1 \parallel s_2) = cf_d(s_1) + cf_d(s_2)$
$cf_d(\int_{t\in T} s) = cf_d(s) + 1$
End 3.2

For each guarded statement g the complexity function cf_d is well-defined. This follows from the definition of the complexity function and the form of guarded statements. In the case of sequential composition only induction with respect to the first argument can be applied. From the above definition we can derive that $cf_d(s\ [r/t]) = cf_d(s)$.

In all proofs below using the induction on the complexity of statements, only the most interesting cases are considered.

The first property states that for all configurations C there is a label l and a configuration $\bar{C}$ such that $C -l\to_d \bar{C}$. Using this property we can show that $\mathcal{O}'_d$ delivers non-empty sets.

Property 3.3
For all $C \in Conf$ there exist $l \in Label$ and $\bar{C} \in Conf_{E\Delta}$ such that $C -l\to_d \bar{C}$.
Proof
We prove that for all $s \in Stat$ and $r \in I\!R_{\geq}$ there exist $l \in Label$ and $\bar{C} \in Conf_{E\Delta}$ such that $[s, r] -l\to_d \bar{C}$, using induction on the complexity of statement s.

1. Let $s \equiv (a, e)$ and $\epsilon = \mathcal{E}(e)$.
   We can distinguish two cases.

(a) For the case that $\epsilon > r$ we have that $[(a,\, e),\, r] -(a,\, \epsilon)\rightarrow_d [E,\, \epsilon]$.

(b) And for the case that $\epsilon \leq r$ we have that $[(a,\, e),\, r] -(\delta,\, r)\rightarrow_d [\Delta,\, r]$.

2. Let $s \equiv x\ (e_1,\, \ldots,\, e_n)$ and $x\ (t_1,\, \ldots,\, t_n) \Leftarrow g \in d$.
   $\exists\ l \in Label\ \exists\ \bar{C} \in Conf_{E\Delta}\ [x\ (e_1,\, \ldots,\, e_n),\, r] -l\rightarrow_d \bar{C}$
   $\Leftrightarrow$
   $\exists\ l \in Label\ \exists\ \bar{C} \in Conf_{E\Delta}\ [g\ [e_1/t_1,\, \ldots,\, e_n/t_n],\, r] -l\rightarrow_d \bar{C}$
   which follows from the induction hypothesis.

3. Let $s \equiv s_1\ ;\ s_2$.
   $\exists\ l \in Label\ \exists\ \bar{C} \in Conf_{E\Delta}\ [s_1\ ;\ s_2,\, r] -l\rightarrow_d \bar{C}$
   $\Leftarrow$ # Let $\bar{C}' \equiv [\bar{s},\, r']$ and take $\bar{C} \equiv [\bar{s}\ ;\ s_2,\, r']$ #
   $\exists\ l \in Label\ \exists\ \bar{C}' \in Conf_{E\Delta}\ [s_1,\, r] -l\rightarrow_d \bar{C}'$
   which follows from the induction hypothesis.

4. Let $s \equiv s_1 \cup s_2$.
   $\exists\ l \in Label\ \exists\ \bar{C} \in Conf_{E\Delta}\ [s_1 \cup s_2,\, r] -l\rightarrow_d \bar{C}$
   $\Leftarrow$
   $\exists\ l \in Label\ \exists\ \bar{C} \in Conf_{E\Delta}\ [s_1,\, r] -l\rightarrow_d \bar{C}$
   which follows from the induction hypothesis.

End 3.3

The second property is used to show that $\mathcal{O}'_d$ is closed. This second property states that for all configurations C and for all labels l the number of configurations $\bar{C}$ such that $C -l\rightarrow_d \bar{C}$ is finite. This corresponds to the notion of image finiteness of labelled transition systems.

Property 3.4
For all $C \in Conf$ and $l \in Label$ the set $\{\ \bar{C} \mid C -l\rightarrow_d \bar{C}\ \}$ is finite.
Proof
We define $S\ (s,\, r,\, l) = \{\ \bar{C} \mid [s,\, r] -l\rightarrow_d \bar{C}\ \}$. We prove that $S\ (s,\, r,\, l)$ is finite for all $s \in Stat$, $r \in I\!R_{\geq}$ and $l \in Label$ using induction on the complexity of statement s.

1. Let $s \equiv (a,\, e)$ and $\epsilon = \mathcal{E}\ (e)$.
   We can distinguish three cases.

   (a) Let $\epsilon > r$ and $l = (a,\, \epsilon)$.
   $S\ ((a,\, e),\, r,\, (a,\, \epsilon))$
   $=$
   $\{\ \bar{C} \mid [(a,\, e),\, r] -(a,\, \epsilon)\rightarrow_d \bar{C}\ \}$
   $=$
   $\{[E,\, \epsilon]\}$

   (b) Let $\epsilon \leq r$ and $l = (\delta,\, r)$.
   $S\ ((a,\, e),\, r,\, (\delta,\, r))$
   $=$
   $\{\ \bar{C} \mid [(a,\, e),\, r] -(\delta,\, r)\rightarrow_d \bar{C}\ \}$
   $=$
   $\{[\Delta,\, r]\}$

   (c) Otherwise $S\ ((a,\, e),\, r,\, l) = \emptyset$.

5. Let $s \equiv s_1 \parallel s_2$.
   $S\ (s_1 \parallel s_2,\, r,\, l)$

$=$
$\{\ \bar{C} \mid [s_1 \parallel s_2,\ r] -l\rightarrow_d \bar{C}\ \}$
$=$
$\{\ [\bar{s} \parallel s_2,\ r'] \mid [s_1,\ r] -l\rightarrow_d [\bar{s},\ r']\ \} \cup \{\ [s_1 \parallel \bar{s},\ r'] \mid [s_2,\ r] -l\rightarrow_d [\bar{s},\ r']\ \}$
$=$
$\{\ [\bar{s} \parallel s_2,\ r'] \mid [\bar{s},\ r'] \in S\ (s_1,\ r,\ l)\ \} \cup \{\ [s_1 \parallel \bar{s},\ r'] \mid [\bar{s},\ r'] \in S\ (s_2,\ r,\ l)\ \}$
This set is finite, because the sets $S\ (s_1,\ r,\ l)$ and $S\ (s_2,\ r,\ l)$ are finite, which follows from the induction hypothesis.

6. Let $s \equiv \int_{t \in T}\ s$.
$S\ (\int_{t \in T}\ s,\ r,\ l)$
$=$
$\{\ \bar{C} \mid [\int_{t \in T}\ s,\ r] -l\rightarrow_d \bar{C}\ \}$
$=$
$\bigcup \{\ \{\ \bar{C} \mid [s\ [r'/t],\ r] -l\rightarrow_d \bar{C}\ \} \mid r' \in T\ \}$
$=$
$\bigcup \{\ S\ (s\ [r'/t],\ r,\ l) \mid r' \in T\ \}$
This set is finite, because the sets $S\ (s\ [r'/t],\ r,\ l)$ are finite, which follows from the induction hypothesis, and T is finite.

End 3.4

Note that the restriction of time sets to finite sets is essential in the proof of this property. We can even prove that the set $\{\ (l,\ \bar{C}) \mid C -l\rightarrow_d \bar{C}\ \}$ is finite for all configurations C, which corresponds to the notion of finitely branching of labelled transition systems.

Applying the mapping $\mathcal{O}_d$ to the configuration $[\int_{t \in I\!N}\ x\ (1,\ t+1),\ 0]$, where $x\ (t_1,\ t_2)$ is declared as $(a,\ min\ \{t_1,\ t_2\})\ ;\ x\ (t_1+1,\ t_2)$, delivers a non-closed set.

Having proved properties 3.3 and 3.4, we can show that $\mathcal{O}'_d$ is well-defined as stated in the following lemma.

Lemma 3.5
The mapping $\mathcal{O}'_d$ is well-defined.
Proof
We have to prove for all $C \in Conf$ that $\mathcal{O}'_d\ (C) \in \mathcal{P}_{nc}\ (TFS')$. The fact $\mathcal{O}'_d\ (C) \in \mathcal{P}\ (TFS')$ follows immediately from the definition of $\mathcal{O}'_d$. From property 3.3 we can conclude that $\mathcal{O}'_d$ delivers non-empty sets. We have left to prove that $\mathcal{O}'_d\ (C)$ is a closed set. Let $\{\sigma_i\}_i$ be a Cauchy sequence in $\mathcal{O}'_d\ (C)$, which converges to σ. We only consider the case that $\sigma \in Label^{\omega}$. Let $\sigma = l_1\ l_2\ \ldots$. We show that there exist configurations $C_1,\ C_2,\ \ldots$ such that $C -l_1\rightarrow_d C_1$ and $C_1 -l_2\rightarrow_d C_2,\ \ldots$. For $\{\sigma_i\}_i$ is a Cauchy sequence, there exists an infinite subsequence $\{\sigma_{f(i)}\}_i$ such that $\sigma_{f(i)} = l_1 \cdot \sigma'_{f(i)}$. From property 3.4 we can deduce that there exists an infinite subsequence $\{\sigma_{g(f(i))}\}_i$ such that $\sigma_{g(f(i))} = l_1 \cdot \sigma'_{g(f(i))}$ such that $C -l_1\rightarrow_d C_1$ for some fixed configuration C_1. We can continue this process ad infinitum. Hence $\sigma \in \mathcal{O}'_d\ (C)$.
End 3.5

We can also define the operational semantics as the unique fixed point, which we call $\mathcal{O}_d$, of a higher-order mapping $\Psi_{\mathcal{O}_d}$. This mapping and the mapping $\mathcal{O}'_d$ give rise to the same sets for each configuration as will be shown in lemma 3.10. The former mapping will be used to prove the equivalence of the operational and the denotational semantics. With abuse of language we will write $\Psi_{\mathcal{O}}$ instead of $\Psi_{\mathcal{O}_d}$. We show that this higher-order mapping is well-defined and that it is a contraction. Due to Banach's theorem, we have that $\Psi_{\mathcal{O}}$ has a unique fixed-point. The higher-order mapping $\Psi_{\mathcal{O}}$ is given in the following definition.

Definition 3.6
The mapping $\Psi_{\mathcal{O}} : (Conf \rightarrow \mathcal{P}_{nc}\ (TFS')) \rightarrow (Conf \rightarrow \mathcal{P}_{nc}\ (TFS'))$ is given by
$\Psi_{\mathcal{O}}\ (F)(C) = \bigcup \{\ l \cdot F\ (C') \mid C -l\rightarrow_d C'\ \} \cup \{\ l \mid C -l\rightarrow_d [E, r]\ \} \cup \{\ l \mid C -l\rightarrow_d [\Delta, r]\ \}$
End 3.6

Now we show that $\Psi_{\mathcal{O}}$ is well-defined.

Lemma 3.7
The mapping $\Psi_{\mathcal{O}}$ is well-defined.
Proof
We prove for all $F \in Conf \rightarrow \mathcal{P}_{nc}\ (TFS')$ and $C \in Conf$ that $\Psi_{\mathcal{O}}\ (F)(C) \in \mathcal{P}_{nc}\ (TFS')$. From the definition of $\Psi_{\mathcal{O}}$ immediately follows that $\Psi_{\mathcal{O}} \in \mathcal{P}\ (TFS')$. By property 3.3, we have that for all configurations C there exists a label l and a configuration $\bar{C}$ such that $C -l\rightarrow_d \bar{C}$. Now immediately follows that the set $\Psi_{\mathcal{O}}$ is non-empty. We have left to prove that this set is also closed. Let $\{\sigma_i\}_i$ be a Cauchy sequence in $\Psi_{\mathcal{O}}\ (F)(C)$. By the definition of $\Psi_{\mathcal{O}}$, there exists an infinite subsequence $\{\sigma_{f(i)}\}_i$ in one of the three following sets.

1. $\bigcup \{\ l \cdot F\ (C') \mid C -l\rightarrow_d C'\ \}$
2. $\{\ l \mid C -l\rightarrow_d [E, r]\ \}$
3. $\{\ l \mid C -l\rightarrow_d [\Delta, r]\ \}$

We only consider the first case. This Cauchy sequence $\{\sigma_{f(i)}\}_i$ is of the form $\{l_{f(i)} \cdot \sigma_{f(i)}\}_i$ where $C -l_{f(i)}\rightarrow_d C'$ for some configuration C'. Since it is a Cauchy sequence there exists an infinite subsequence $\{l \cdot \sigma_{g(f(i))}\}_i$ of $\{l_{f(i)} \cdot \sigma_{f(i)}\}_i$, for some fixed label l such that $C -l\rightarrow_d C'$ for some configuration C'. By property 3.4 we have that for all configurations C and all labels l there are only a finite number configurations C' such that $C -l\rightarrow_d C'$. So by the pigeon-hole principle, there exists an infinite subsequence of $\{l \cdot \sigma_{g(f(i))}\}_i$ for some fixed label l and configuration C' such that $C -l\rightarrow_d C'$. Since $F\ (C')$ is closed, this subsequence converges to an element in $l \cdot F\ (C')$, which is also an element of $\Psi_{\mathcal{O}}\ (F)(C)$. So the whole sequence converges to that same element.
End 3.7

Next we show that $\Psi_{\mathcal{O}}$ is a contraction.

Lemma 3.8
The mapping $\Psi_{\mathcal{O}}$ is a contraction.
Proof
$d\ (\Psi_{\mathcal{O}}\ (F)(C), \Psi_{\mathcal{O}}\ (G)(C))$
$=$
$d\ (\bigcup \{\ l \cdot F\ (C') \mid C -l\rightarrow_d C'\ \} \cup \{\ l \mid C -l\rightarrow_d [E, r]\ \} \cup \{\ l \mid C -l\rightarrow_d [\Delta, r]\ \},$
$\quad \bigcup \{\ l \cdot G\ (C') \mid C -l\rightarrow_d C'\ \} \cup \{\ l \mid C -l\rightarrow_d [E, r]\ \} \cup \{\ l \mid C -l\rightarrow_d [\Delta, r]\ \})$
$\leq$ # $d\ (l \cdot F\ (C'), l \cdot G\ (C')) \leq \frac{1}{2}\ d\ (F, G)$, property A.11 of the appendix # $\frac{1}{2}\ d\ (F, G)$
End 3.8

Now we know that $\Psi_{\mathcal{O}}$ has a unique fixed-point. Using the fixed point property gives us the following definition of $\mathcal{O}_d$.

Definition 3.9
The mapping $\mathcal{O}_d : Conf \rightarrow \mathcal{P}_{nc}\ (TFS')$ is given by
$\mathcal{O}_d\ (C) = \bigcup \{\ l \cdot \mathcal{O}_d\ (C') \mid C -l\rightarrow_d C'\ \} \cup \{\ l \mid C -l\rightarrow_d [E, r]\ \} \cup \{\ l \mid C -l\rightarrow_d [\Delta, r]\ \}$
End 3.9

As already mentioned, we show that $\mathcal{O}'_d$ and $\mathcal{O}_d$ give rise to the same sets for each configuration. We do this by showing that $\mathcal{O}'_d$ is a fixed point of $\Psi_{\mathcal{O}}$.

Lemma 3.10
$\mathcal{O}'_d = \Psi_{\mathcal{O}}\ (\mathcal{O}'_d)$
Proof
$\sigma \in \mathcal{O}'_d\ (C)$
$\Leftrightarrow$
$\exists\ C' \in Conf\ \exists\ l \in Label\ C \mathrel{-}l\mathrel{\rightarrow}_d C' \wedge \sigma = l\ \sigma' \wedge \sigma' \in \mathcal{O}'_d\ (C') \vee$
$\exists\ r \in I\!R_{\geq}\ C \mathrel{-}l\mathrel{\rightarrow}_d [E,\ r] \vee$
$\exists\ r \in I\!R_{\geq}\ C \mathrel{-}l\mathrel{\rightarrow}_d [\Delta,\ r]$
$\Leftrightarrow$
$\sigma \in \bigcup \{\ l \cdot F\ (C') \mid C \mathrel{-}l\mathrel{\rightarrow}_d C'\ \} \cup \{\ l \mid C \mathrel{-}l\mathrel{\rightarrow}_d [E,\ r]\ \} \cup \{\ l \mid C \mathrel{-}l\mathrel{\rightarrow}_d [\Delta,\ r]\ \}$
$\Leftrightarrow$
$\sigma \in \Psi_{\mathcal{O}}\ (\mathcal{O}'_d)(C)$
End 3.10

In the following we restrict our semantic domain. The set TFS' is restricted to the set of timed failure streams TFS to exclude meaningless streams.

$TFS = \{\ \sigma \in TA^+ \cup TA^*.TF \cup TA^\omega \mid$
if $(a,\ r)$ precedes $(a',\ r')$ in σ then $r < r'$,
if $(a,\ r)$ immediately precedes $(\delta,\ r')$ in σ then $r = r'\ \}$
with typical element σ.

The fact that if $(a,\ r)$ precedes $(a',\ r')$ in σ then $r < r'$ represents that two successive atomic actions cannot be executed at the same time. The clause if $(a,\ r)$ immediately precedes $(\delta,\ r')$ in σ then $r = r'$ represents the notion of failure.

To justify this restriction, we show that for each configuration the mapping $\mathcal{O}'_d$ delivers an element of $\mathcal{P}\ (TFS)$. This follows immediately from the properties 3.11 and 3.12. The first property shows that if $(a,\ r)$ precedes $(a',\ r')$ then $r < r'$.

Property 3.11
For all $s \in Stat$, $\bar{s} \in Stat_{E\Delta}$, $a \in A$ and $r,\ r',\ r'' \in I\!R_{\geq}$
$[s,\ r] \mathrel{-}(a,\ r')\mathrel{\rightarrow}_d [\bar{s},\ r''] \Rightarrow r' = r'' \wedge r < r'$
Proof
We prove this using induction on the complexity of statement s.

1. Let $s \equiv (a,\ e)$ and $\epsilon = \mathcal{E}\ (e)$. Since the only rule which satisfies the premise is $[(a,\ e),\ r] \mathrel{-}(a,\ \epsilon)\mathrel{\rightarrow}_d [E,\ \epsilon]$, where $r < \epsilon$, it is clear that the property holds in this case.

2. Let $s \equiv x\ (e_1,\ \ldots,\ e_n)$ and $x\ (t_1,\ \ldots,\ t_n) \Leftarrow g \in d$.
   $[x\ (e_1,\ \ldots,\ e_n),\ r] \mathrel{-}(a,\ r')\mathrel{\rightarrow}_d [\bar{s},\ r'']$
   $\Leftrightarrow$
   $[g\ [e_1/t_1,\ \ldots,\ e_n/t_n],\ r] \mathrel{-}(a,\ r')\mathrel{\rightarrow}_d [\bar{s},\ r'']$
   $\Rightarrow$
   $r' = r'' \wedge r < r'$

5. Let $s \equiv s_1 \parallel s_2$.
   $[s_1 \parallel s_2,\ r] \mathrel{-}(a,\ r')\mathrel{\rightarrow}_d [\bar{s},\ r'']$
   $\Leftrightarrow$
   $([s_1,\ r] \mathrel{-}(a,\ r')\mathrel{\rightarrow}_d [\bar{s}',\ r''] \wedge \bar{s} \equiv \bar{s}' \parallel s_2) \vee ([s_2,\ r] \mathrel{-}(a,\ r')\mathrel{\rightarrow}_d [\bar{s},\ r''] \wedge \bar{s} \equiv s_1 \parallel \bar{s}')$
   $\Rightarrow$
   $r' = r'' \wedge r < r'$

End 3.11

The second property shows that if a failure occurs in a stream, it will be the last element of that stream. In conjunction with the first property it also shows that if $(a,\ r)$ immediately precedes $(\delta,\ r')$ then $r = r'$.

Property 3.12
For all $s \in Stat$, $\bar{s} \in Stat_{E\Delta}$ and $r,\ r',\ r'' \in I\!R_{\geq}$
$[s,\ r]\ -(\delta,\ r')\rightarrow_d\ [\bar{s},\ r'']\ \Rightarrow\ r = r' \wedge r' = r'' \wedge \bar{s} = \Delta$
Proof
We prove this using induction on the complexity of statement s.

1. Let $s \equiv (a,\ e)$ and $\epsilon = \mathcal{E}\ (e)$. Since the only rule which satisfies the premise is $[(a,\ e),\ r]\ -(\delta,\ r)\rightarrow_d\ [\Delta,\ r]$, where $r \geq \epsilon$, it is clear that the property holds in this case.

3. Let $s \equiv s_1\ ;\ s_2$.
   $[s_1\ ;\ s_2,\ r]\ -(\delta,\ r')\rightarrow_d\ [\bar{s},\ r'']$
   $\Leftrightarrow$
   $[s_1,\ r]\ -(\delta,\ r')\rightarrow_d\ [\bar{s}',\ r''] \wedge \bar{s} \equiv \bar{s}'\ ;\ s_2$
   $\Rightarrow \qquad \#\ \Delta\ ;\ s_2 = \Delta\ \#$
   $r = r' \wedge r' = r'' \wedge \bar{s} = \Delta$

6. Let $s \equiv \int_{t \in T}\ s$.
   $[\int_{t \in T}\ s,\ r]\ -(\delta,\ r')\rightarrow_d\ [\bar{s},\ r'']$
   $\Leftrightarrow$
   $\exists\ \bar{r} \in T\ [s\ [\bar{r}/t],\ r]\ -(\delta,\ r')\rightarrow_d\ [\bar{s},\ r'']$
   $\Rightarrow$
   $r = r' \wedge r' = r'' \wedge \bar{s} = \Delta$

End 3.12

We conclude this section with the definition of the operational semantics for programs. The well-definedness of the operational semantics $\mathcal{O}$ follows from the well-definedness of $\mathcal{O}'_d$.

Definition 3.13
The mapping $\mathcal{O} : Prog \rightarrow \mathcal{P}_{nc}\ (TFS)$ is given by
$\mathcal{O}\ (d \mid s) = \mathcal{O}'_d\ ([s,\ 0])$
End 3.13

4 Denotational semantics

After having defined an operational semantics, we give a denotational semantics for our language. A denotational semantics $\mathcal{D}$ should be compositional, i.e. for every syntactic operator *op* a corresponding semantics operator *op'* should be defined such that

$\mathcal{D}\ (d \mid s_1\ op\ s_2) = \mathcal{D}\ (d \mid s_1)\ op'\ \mathcal{D}\ (d \mid s_2)$

and it should tackle recursion with the help of fixed points.

This denotational semantics does not record failures, but just streams of timed actions, which are not even ordered in time.

The semantic domain used to define the denotational semantics is the set of timed streams

$TS = TA^+ \cup TA^\omega$

with typical element σ.

For the syntactic operator ; we define the corresponding semantic operator $\bullet$. We first define this operator on timed streams. Then we extend this definition to non-empty closed sets of timed streams.

Definition 4.1
The operator $\bullet : TS \times TS \rightarrow TS$ is given by
$\alpha \bullet \tau = \alpha \cdot \tau$
$(\alpha \cdot \sigma) \bullet \tau = \alpha \cdot (\sigma \bullet \tau)$
The operator $\bullet : \mathcal{P}_{nc}(TS) \times \mathcal{P}_{nc}(TS) \rightarrow \mathcal{P}_{nc}(TS)$ is given by
$\Sigma \bullet \mathcal{T} = \{ \sigma \bullet \tau \mid \sigma \in \Sigma \wedge \tau \in \mathcal{T} \}$
End 4.1

It may not be obvious that the operator $\bullet$ is well-defined.

Lemma 4.2
The operator $\bullet$ is well-defined.
Proof
We have to prove that the set $\Sigma \bullet \mathcal{T}$ is closed. Let $\{\rho_i\}_i$ be a Cauchy sequence in $\Sigma \bullet \mathcal{T}$. So $\rho_i = \sigma_i \bullet \tau_i$ for some $\sigma_i \in \Sigma$ and $\tau_i \in \mathcal{T}$. Let $n_1 \in I\!N$. Since $\{\rho_i\}_i$ is a Cauchy sequence, there exists a subsequence $\{\rho_{f_1(i)}\}_i$ such that $\rho_{f_1(i)}\ [n_1]$, the prefixes of $\rho_{f_1(i)}$ of length n_1, are constant. By the pigeon-hole principle we have that there exists a subsequence $\{\rho_{g_1(f_1(i))}\}_i$ such that $\rho_{g_1(f_1(i))}\ [n_1] = \sigma \bullet \tau_{g_1(f_1(i))}\ [n_1']$, where $\sigma \in \Sigma$, $\tau_{g_1(f_1(i))} \in \mathcal{T}$ and $n_1' = n_1$ - length (σ), or $\rho_{g_1(f_1(i))}\ [n_1] = \sigma_{g_1(f_1(i))}\ [n_1]$, where $\sigma_{g_1(f_1(i))} \in \Sigma$. In the first case $\rho_{g_1(f_1(i))} = \sigma \bullet \tau_{g_1(f_1(i))}$. The Cauchy sequence $\{\tau_{g_1(f_1(i))}\}_i$ converges to some $\tau \in \mathcal{T}$, due to the fact that $\mathcal{T}$ is closed. So $\{\rho_{g_1(f_1(i))}\}_i$ converges to $\sigma \bullet \tau$. From this we can conclude that the sequence $\{\rho_i\}_i$ also converges to $\sigma \bullet \tau$, which is an element of $\Sigma \bullet \mathcal{T}$. In the second case, we can continue the process by taking a subsequence $\{\rho_{f_2(g_1(f_1(i)))}\}_i$, whose prefixes of length $n_2 > n_1$ are constant. If the process terminates, we will end up in the first case, which gives us the desired result. If the process does not terminate, we define $h(i) = g_i(f_i(\ldots g_1(f_1(0))\ldots))$. It is easy to see that $\{\rho_{h(i)}\}_i$ is a Cauchy sequence in Σ, which converges to some $\sigma \in \Sigma$, since Σ is closed. This σ is infinite and therefore an element of $\Sigma \bullet \mathcal{T}$. So we can conclude that $\{\rho_i\}_i$ converges to an element of $\Sigma \bullet \mathcal{T}$.
End 4.2

We can also define the operator $\bullet$ as a fixed point of a higher-order mapping. We therefore first introduce an auxiliary operator.

Definition 4.3
For each $\Sigma \in \mathcal{P}_{nc}(TS)$ and $\alpha \in TA$, $\Sigma_\alpha \in \mathcal{P}_c(TS)$ is given by
$\Sigma_\alpha = \{\sigma \in TS \mid \alpha \cdot \sigma \in \Sigma\}$
End 4.3

It is straightforwardly shown that this auxiliary operator is well-defined.

Lemma 4.4
For all $\Sigma \in \mathcal{P}_{nc}(TS)$ and $\alpha \in TA$, $\Sigma_\alpha \in \mathcal{P}_c(TS)$
Proof
Σ_α is an element of $\mathcal{P}(TS)$, which follows from the definition. Let $\{\sigma_i\}_i$ be a Cauchy sequence in Σ_α. Then $\{\alpha \cdot \sigma_i\}_i$ is a Cauchy sequence in Σ. Since Σ is closed, the Cauchy sequence $\{\alpha \cdot \sigma_i\}_i$ converges to an element of Σ , which will be of the form $\alpha \cdot \sigma$. So the Cauchy sequence $\{\sigma_i\}_i$ will converge to σ, which is an element of Σ_α.
End 4.4

Now we can define the operator the higher-order mapping as follows.

Definition 4.5
The mapping $\Psi_\bullet : (\mathcal{P}_{nc}(TS) \times \mathcal{P}_{nc}(TS) \rightarrow \mathcal{P}_{nc}(TS)) \rightarrow (\mathcal{P}_{nc}(TS) \times \mathcal{P}_{nc}(TS) \rightarrow \mathcal{P}_{nc}(TS))$ is given by

$\Psi_\bullet\ (F)(\Sigma,\ \mathcal{T}) = \bigcup \{\ \alpha\ \cdot\ F(\Sigma_\alpha,\ \mathcal{T})\ |\ \Sigma_\alpha \neq \emptyset\ \} \cup \bigcup \{\ \alpha\ \cdot\ \mathcal{T}|\ \alpha \in \Sigma\ \}$
End 4.5

We show that $\Psi_\bullet$ is well-defined and that it is a contraction.

Lemma 4.6
The mapping $\Psi_\bullet$ is well-defined.
Proof
We have to prove for all $F \in (\mathcal{P}_{nc}\ (TS) \times \mathcal{P}_{nc}\ (TS)) \rightarrow \mathcal{P}_{nc}\ (TS)$ and $\Sigma,\ \mathcal{T} \in \mathcal{P}_{nc}\ (TS)$ that $\Psi_\bullet\ (F)(\Sigma,\ \mathcal{T}) \in \mathcal{P}_{nc}\ (TS)$. The fact that $\Psi_\bullet\ (F)(\Sigma,\ \mathcal{T}) \in \mathcal{P}\ (TS)$ follows from the definition. Since Σ is non-empty, there must be an α such that Σ_α is non-empty or $\alpha \in \Sigma$, which gives us that $\Psi_\bullet\ (F)(\Sigma,\ \mathcal{T})$ is non-empty. We have left to prove that $\Psi_\bullet\ (F)(\Sigma,\ \mathcal{T})$ is closed. Let $\{\sigma_i\}_i$ be a Cauchy sequence in $\Psi_\bullet\ (F)(\Sigma,\ \mathcal{T})$. Then there exists an infinite subsequence $\{\sigma_{f(i)}\}_i$ in one of the following sets.

1. $\bigcup \{\ \alpha\ \cdot\ F\ (\Sigma_\alpha, \mathcal{T})\ |\ \Sigma_\alpha \neq \emptyset\ \}$
2. $\bigcup \{\ \alpha\ \cdot\ \mathcal{T}|\ \alpha \in \Sigma\ \}$

We only consider the first case. In this case there exists an infinite subsequence $\{\sigma_{g(f(i))}\}_i$ in $\alpha\ \cdot\ F\ (\Sigma_\alpha,\ \mathcal{T})$ for a fixed α. Since $F\ (\Sigma_\alpha,\ \mathcal{T})$ is closed, the sequence $\{\sigma_{g(f(i))}\}_i$ converges to an element in $\alpha\ \cdot\ F\ (\Sigma_\alpha,\ \mathcal{T})$. So the whole Cauchy sequence converges to that same element, which is also an element of $\Psi_\bullet\ (F)(\Sigma,\ \mathcal{T})$.
End 4.6

Lemma 4.7
The mapping $\Psi_\bullet$ is a contraction.
Proof
$d\ (\Psi_\bullet\ (F)(\Sigma,\ \mathcal{T}),\ \Psi_\bullet\ (G)(\Sigma,\ \mathcal{T}))$
$=$
$d\ (\bigcup \{\ \alpha\ \cdot\ F\ (\Sigma_\alpha,\ \mathcal{T})\ |\ \Sigma_\alpha \neq \emptyset\ \} \cup \bigcup \{\ \alpha\ \cdot\ \mathcal{T}|\ \alpha \in \Sigma\ \},$
$\quad \bigcup \{\ \alpha\ \cdot\ G\ (\Sigma_\alpha,\ \mathcal{T})\ |\ \Sigma_\alpha \neq \emptyset\ \} \cup \bigcup \{\ \alpha\ \cdot\ \mathcal{T}|\ \alpha \in \Sigma\ \})$
$\leq$ # property A.11 #
$max\ (d\ (\bigcup \{\ \alpha\ \cdot\ F\ (\Sigma_\alpha,\ \mathcal{T})\ |\ \Sigma_\alpha \neq \emptyset\ \}, \bigcup \{\ \alpha\ \cdot\ G\ (\Sigma_\alpha,\ \mathcal{T})\ |\ \Sigma_\alpha \neq \emptyset\ \}),$
$\quad d\ (\bigcup \{\ \alpha\ \cdot\ \mathcal{T}|\ \alpha \in \Sigma\ \}, \bigcup \{\ \alpha\ \cdot\ \mathcal{T}|\ \alpha \in \Sigma\ \}))$
$=$
$d\ (\bigcup \{\ \alpha\ \cdot\ F\ (\Sigma_\alpha,\ \mathcal{T})\ |\ \Sigma_\alpha \neq \emptyset\ \}, \bigcup \{\ \alpha\ \cdot\ G\ (\Sigma_\alpha,\ \mathcal{T})\ |\ \Sigma_\alpha \neq \emptyset\ \})$
$\leq$ # $\Sigma_\alpha \neq \emptyset \Rightarrow d\ (\alpha\ \cdot\ F\ (\Sigma_\alpha,\ \mathcal{T})\ \alpha\ \cdot\ G\ (\Sigma_\alpha,\ \mathcal{T})) \leq \frac{1}{2}\ d\ (F,\ G)$, property A.11 #
$\frac{1}{2}\ d\ (F,\ G)$
End 4.7

Because $\Psi_\bullet$ is a contraction we know that $\Psi_\bullet$ has a unique fixed point according to Banach's theorem. The fixed point property gives us the following definition for the operator $\bullet$.

Definition 4.8
The operator $\bullet : \mathcal{P}_{nc}\ (TS) \times \mathcal{P}_{nc}\ (TS) \rightarrow \mathcal{P}_{nc}\ (TS)$ is given by
$\Sigma \bullet \mathcal{T} = \bigcup \{\ \alpha\ \cdot\ (\Sigma_\alpha \bullet \mathcal{T})\ |\ \Sigma_\alpha \neq \emptyset\ \} \cup \bigcup \{\ \alpha\ \cdot\ \mathcal{T}|\ \alpha \in \Sigma\ \}$
End 4.8

We can easily verify that the operators defined in the definitions 4.1 and 4.8 are equivalent, because $\Psi_\bullet\ (\bullet)(\Sigma,\ \mathcal{T}) = \Sigma \bullet \mathcal{T}$.

Next we define two semantic operators which correspond to the syntactic notions of non-deterministic choice and integration.

Definition 4.9
The operator $\cup : \mathcal{P}_{nc}(TS) \times \mathcal{P}_{nc}(TS) \to \mathcal{P}_{nc}(TS)$ is defined as the set-theoretic union.
The operator $\bigcup : (\mathcal{P}_{nc}(TS))^* \to \mathcal{P}_{nc}(TS)$ is defined as the generalised set-theoretic union.
End 4.9

The semantic counterpart of the syntactic operator $\|$ is first defined on timed streams. This is done by defining a so-called left-merge $\lfloor\!\lfloor$, as introduced by Bergstra and Klop in [BK82], which expresses a merge where the first element is taken from the left argument. Then we extend this definition to non-empty closed sets of timed streams.

Definition 4.10
The operator $\| : TS \times TS \to \mathcal{P}(TS)$ is given by
$\sigma \| \tau = \sigma \lfloor\!\lfloor \tau \cup \tau \lfloor\!\lfloor \sigma$
The operator $\lfloor\!\lfloor : TS \times TS \to \mathcal{P}(TS)$ is given by
$\alpha \lfloor\!\lfloor \tau = \{\alpha \cdot \tau\}$
$(\alpha \cdot \sigma) \lfloor\!\lfloor \tau = \alpha \cdot (\sigma \| \tau)$
The operator $\| : \mathcal{P}_{nc}(TS) \times \mathcal{P}_{nc}(TS) \to \mathcal{P}_{nc}(TS)$ is given by
$\Sigma \| \mathcal{T} = \bigcup \{ \sigma \| \tau \mid \sigma \in \Sigma \wedge \tau \in \mathcal{T} \}$
The operator $\lfloor\!\lfloor : \mathcal{P}_{nc}(TS) \times \mathcal{P}_{nc}(TS) \to \mathcal{P}_{nc}(TS)$ is given by
$\Sigma \lfloor\!\lfloor \mathcal{T} = \bigcup \{ \sigma \lfloor\!\lfloor \tau \mid \sigma \in \Sigma \wedge \tau \in \mathcal{T} \}$
End 4.10

Proving that the operator $\|$ is well-defined can be done along the lines of the well-definedness proof of the operator $\bullet$ and is left to the reader.

Again we can characterise the operator as the unique fixed point of a higher-order mapping.

Definition 4.11
The mapping $\Psi_\| : (\mathcal{P}_{nc}(TS) \times \mathcal{P}_{nc}(TS) \to \mathcal{P}_{nc}(TS)) \to (\mathcal{P}_{nc}(TS) \times \mathcal{P}_{nc}(TS) \to \mathcal{P}_{nc}(TS))$ is given by
$\Psi_\|(F)(\Sigma, \mathcal{T}) = \Psi_\bullet(F)(\Sigma, \mathcal{T}) \cup \Psi_\bullet(F)(\mathcal{T}, \Sigma)$
End 4.11

So the fixed point property gives us the following definition, which is equivalent to the earlier given definition of $\|$.

Definition 4.12
The operator $\| : \mathcal{P}_{nc}(TS) \times \mathcal{P}_{nc}(TS) \to \mathcal{P}_{nc}(TS)$ is given by
$$\Sigma \| \mathcal{T} = \bigcup \{ \alpha \cdot (\Sigma_\alpha \| \mathcal{T}) \mid \Sigma_\alpha \neq \emptyset \} \cup \bigcup \{ \alpha \cdot \mathcal{T} \mid \alpha \in \Sigma \} \cup \bigcup \{ \alpha \cdot (\mathcal{T}_\alpha \| \Sigma) \mid \mathcal{T}_\alpha \neq \emptyset \} \cup \bigcup \{ \alpha \cdot \Sigma \mid \alpha \in \mathcal{T} \}$$
End 4.12

Now we can give the denotational semantics for statements recorded in the mapping $\mathcal{D}_d$, which is the fixed point of the higher-order mapping $\Psi_\mathcal{D}$.

Definition 4.13
The mapping $\mathcal{D}_d : Stat \to \mathcal{P}_{nc}(TS)$ is given by
$\mathcal{D}_d((a, e)) = \{(a, \epsilon)\}$ $\qquad \mathcal{E}(e) = \epsilon$
$\mathcal{D}_d(x(e_1, \ldots, e_n)) = \mathcal{D}_d(g[e_1/t_1, \ldots, e_n/t_n])$ $\qquad x(t_1, \ldots, t_n) \Leftarrow g \in d$
$\mathcal{D}_d(s_1 ; s_2) = \mathcal{D}_d(s_1) \bullet \mathcal{D}_d(s_2)$
$\mathcal{D}_d(s_1 \cup s_2) = \mathcal{D}_d(s_1) \cup \mathcal{D}_d(s_2)$
$\mathcal{D}_d(s_1 \| s_2) = \mathcal{D}_d(s_1) \| \mathcal{D}_d(s_2)$
$\mathcal{D}_d(\int_{t \in T} s) = \bigcup \{\mathcal{D}_d(s[r/t]) \mid r \in T\}$
End 4.13

The higher-order mapping $\Psi_\mathcal{D}$ is defined in the following definition.

Definition 4.14
The mapping $\Psi_{\mathcal{D}} : (Stat \to \mathcal{P}_{nc}\ (TS)) \to (Stat \to \mathcal{P}_{nc}\ (TS))$ is given by
$\Psi_{\mathcal{D}}\ (F)\ ((a,\ e)) = \{(a,\ \epsilon)\}$ $\qquad \mathcal{E}\ (e) = \epsilon$
$\Psi_{\mathcal{D}}\ (F)\ (x\ (e_1,\ \ldots,\ e_n)) = \Psi_{\mathcal{D}}\ (F)\ (g\ [e_1/t_1,\ \ldots,\ e_n/t_n])$ $\qquad x\ (t_1,\ \ldots,\ t_n) \Leftarrow g \in d$
$\Psi_{\mathcal{D}}\ (F)\ (s_1\ ;\ s_2) = \Psi_{\mathcal{D}}\ (F)\ (s_1) \bullet F\ (s_2)$
$\Psi_{\mathcal{D}}\ (F)\ (s_1 \cup s_2) = \Psi_{\mathcal{D}}\ (F)\ (s_1) \cup \Psi_{\mathcal{D}}\ (F)\ (s_2)$
$\Psi_{\mathcal{D}}\ (F)\ (s_1 \parallel s_2) = \Psi_{\mathcal{D}}\ (F)\ (s_1) \parallel \Psi_{\mathcal{D}}\ (F)\ (s_2)$
$\Psi_{\mathcal{D}}\ (F)\ (\int_{t \in T} s) = \bigcup \{\Psi_{\mathcal{D}}\ (F)\ (s\ [r/t]) \mid r \in T\}$
End 4.14

The fact that $\Psi_{\mathcal{D}}$ is well-defined follows from the fact that the operators are well-defined. In order to prove that $\Psi_{\mathcal{D}}$ is a contraction we need two properties. The first property states that the operator $\bullet$ is non-distance increasing in its first argument and contracting with factor $\frac{1}{2}$ in its second argument.

Property 4.15
For all $\Sigma,\ \Sigma',\ \mathcal{T},\ \mathcal{T}' \in \mathcal{P}_{nc}\ (TS)$ and $\varepsilon \in [0,\ 1]$
$d\ (\Sigma,\ \Sigma') \leq \varepsilon \wedge d\ (\mathcal{T},\ \mathcal{T}') \leq 2\ \varepsilon \Rightarrow d\ (\Sigma \bullet \mathcal{T},\ \Sigma' \bullet \mathcal{T}') \leq \varepsilon$
Proof
$d\ (\Sigma,\ \Sigma') \leq \varepsilon \wedge d\ (\mathcal{T},\ \mathcal{T}') \leq 2\ \varepsilon$
$\Rightarrow$ # note 1 #
$\forall\ n \geq 0\ d\ (\Psi^n_\bullet\ (F)(\Sigma,\ \mathcal{T}),\ \Psi^n_\bullet\ (F)(\Sigma',\ \mathcal{T}')) \leq \varepsilon$
$\Rightarrow$ # note 2 #
$d\ (lim_n\ \Psi^n_\bullet\ (F)(\Sigma,\ \mathcal{T}),\ lim_n\ \Psi^n_\bullet\ (F)(\Sigma',\ \mathcal{T}')) \leq \varepsilon$
$\Rightarrow$ # Banach's theorem #
$d\ (\Sigma \bullet \mathcal{T},\ \Sigma' \bullet \mathcal{T}') \leq \varepsilon$

Note 1:
For $n = 0$ we have that $d\ (\Sigma,\ \Sigma') \leq \varepsilon \wedge d\ (\mathcal{T},\ \mathcal{T}') \leq 2\ \varepsilon \Rightarrow d\ (F\ (\Sigma,\ \mathcal{T}),\ F\ (\Sigma',\ \mathcal{T}')) \leq \varepsilon$, by taking $F\ (\Sigma,\ \mathcal{T}) = \Sigma$. Assume that $n \geq 0$ and $d\ (\Sigma,\ \Sigma') \leq \varepsilon$ and $d\ (\mathcal{T},\ \mathcal{T}') \leq 2\ \varepsilon$ and
$d\ (\Psi^n_\bullet\ (F)(\Sigma,\ \mathcal{T}),\ \Psi^n_\bullet\ (F)(\Sigma',\ \mathcal{T}')) \leq \varepsilon$. We have to prove that
$d\ (\Psi^{n+1}_\bullet\ (F)(\Sigma,\ \mathcal{T}),\ \Psi^{n+1}_\bullet\ (F)(\Sigma',\ \mathcal{T}')) \leq \varepsilon$
$\Leftrightarrow$
$d\ (\bigcup \{\ \alpha \cdot \Psi^n_\bullet\ (F)(\Sigma_\alpha,\ \mathcal{T}) \mid \Sigma_\alpha \neq \emptyset\ \} \cup \bigcup \{\ \alpha \cdot \mathcal{T} \mid \alpha \in \Sigma\ \},$
$\quad \bigcup \{\ \alpha \cdot \Psi^n_\bullet\ (F)(\Sigma'_\alpha,\ \mathcal{T}') \mid \Sigma'_\alpha \neq \emptyset\ \} \cup \bigcup \{\ \alpha \cdot \mathcal{T}' \mid \alpha \in \Sigma'\ \}) \leq \varepsilon$

This can be proved using property A.11 and the following facts. We distinguish three cases for each $\alpha \in TA$.

1. Assume that $(\alpha \in \Sigma \Leftrightarrow \alpha \in \Sigma') \wedge (\Sigma_\alpha \neq \emptyset \Leftrightarrow \Sigma'_\alpha \neq \emptyset)$.
   $d\ (\Sigma,\ \Sigma') \leq \varepsilon \wedge d\ (\mathcal{T},\ \mathcal{T}') \leq 2\ \varepsilon$
   $\Rightarrow$
   $\Sigma_\alpha \neq \emptyset \wedge \Sigma'_\alpha \neq \emptyset \Rightarrow d\ (\Sigma_\alpha,\ \Sigma'_\alpha) \leq 2\ \varepsilon \wedge d\ (\mathcal{T},\ \mathcal{T}') \leq 2\ \varepsilon$
   $\Rightarrow$
   $\Sigma_\alpha \neq \emptyset \wedge \Sigma'_\alpha \neq \emptyset \Rightarrow d\ (F\ (\Sigma_\alpha,\ \mathcal{T}),\ F\ (\Sigma'_\alpha,\ \mathcal{T}')) \leq 2\ \varepsilon \wedge d\ (\mathcal{T},\ \mathcal{T}') \leq 2\ \varepsilon$
   $\Rightarrow$
   $\Sigma_\alpha \neq \emptyset \wedge \Sigma'_\alpha \neq \emptyset \Rightarrow d\ (\alpha \cdot F\ (\Sigma_\alpha,\ \mathcal{T}),\ \alpha \cdot F\ (\Sigma'_\alpha,\ \mathcal{T}')) \leq \varepsilon \wedge d\ (\alpha \cdot \mathcal{T},\ \alpha \cdot \mathcal{T}') \leq \varepsilon$
2. Assume that $(\alpha \in \Sigma \not\Leftrightarrow \alpha \in \Sigma')$.
   $d\ (\Sigma,\ \Sigma') \leq \varepsilon \wedge d\ (\mathcal{T},\ \mathcal{T}') \leq 2\ \varepsilon$
   $\Rightarrow$ # $d\ (\Sigma,\ \Sigma') = 1$, so $\varepsilon \geq 1$ #
   $\Sigma_\alpha \neq \emptyset \wedge \Sigma'_\alpha \neq \emptyset \Rightarrow d\ (\alpha \cdot F\ (\Sigma_\alpha,\ \mathcal{T}),\ \alpha \cdot F\ (\Sigma'_\alpha,\ \mathcal{T}')) \leq \varepsilon \wedge d\ (\alpha \cdot \mathcal{T},\ \alpha \cdot \mathcal{T}') \leq \varepsilon$
3. Assume that $(\Sigma_\alpha \neq \emptyset \not\Leftrightarrow \Sigma'_\alpha \neq \emptyset) \wedge (\alpha \in \Sigma \Leftrightarrow \alpha \in \Sigma')$. Let $\Sigma_\alpha \neq \emptyset$ and $\Sigma'_\alpha = \emptyset$. Then $\alpha \in \Sigma'$, because $\Sigma' \neq \emptyset$.

$d\ (\Sigma,\ \Sigma') \leq \varepsilon \wedge d\ (\mathcal{T},\ \mathcal{T}') \leq 2\ \varepsilon$
$\Rightarrow$ # $d\ (\Sigma,\ \Sigma') \geq \frac{1}{2}$, so $\varepsilon \geq \frac{1}{2}$ #
$\Sigma_\alpha \neq \emptyset \Rightarrow d\ (\alpha\ \cdot\ F\ (\Sigma_\alpha,\ \mathcal{T}),\ \alpha\ \cdot\ \mathcal{T}') \leq \varepsilon \wedge d\ (\alpha\ \cdot\ \mathcal{T},\ \alpha\ \cdot\ \mathcal{T}') \leq \varepsilon$

Note 2:
Since $\Psi_\bullet$ is contracting with factor $\frac{1}{2}$, $\{\Psi^n_\bullet\ (F)(\Sigma,\ \mathcal{T})\}_n$ and $\{\Psi^n_\bullet\ (F)(\Sigma',\ \mathcal{T}')\}_n$ are Cauchy sequences. For some arbitrary $\eta > 0$, we have that
$d\ (lim_n\ \Psi^n_\bullet\ (F)(\Sigma,\ \mathcal{T}),\ lim_n\ \Psi^n_\bullet\ (F)(\Sigma',\ \mathcal{T}'))$
$\leq$
$d\ (lim_n\ \Psi^n_\bullet\ (F)(\Sigma,\ \mathcal{T}),\ \Psi^m_\bullet\ (F)(\Sigma,\ \mathcal{T})) + d\ (\Psi^m_\bullet\ (F)(\Sigma,\ \mathcal{T}),\ \Psi^m_\bullet\ (F)(\Sigma',\ \mathcal{T}')) +$
$d\ (\Psi^m_\bullet\ (F)(\Sigma',\ \mathcal{T}'),\ lim_n\ \Psi^n_\bullet\ (F)(\Sigma',\ \mathcal{T}'))$
$<$
$\frac{\eta}{2} + \varepsilon + \frac{\eta}{2}$
End 4.15

The second property states that the operator $\|$ is non-distance increasing in both arguments. The proof of this property, which has the same structure as the proof of the preceding property, is left to the reader.

Property 4.16
For all $\Sigma,\ \Sigma',\ \mathcal{T},\ \mathcal{T}' \in \mathcal{P}_{nc}\ (TS)$ and $\varepsilon \in [0,\ 1]$
$d\ (\Sigma,\ \Sigma') \leq \varepsilon \wedge d\ (\mathcal{T},\ \mathcal{T}') \leq \varepsilon \Rightarrow d\ (\Sigma \parallel \mathcal{T},\ \Sigma' \parallel \mathcal{T}') \leq \varepsilon$
End 4.16

Now we are ready to prove the contractivity of $\Psi_\mathcal{D}$.

Lemma 4.17
The mapping $\Psi_\mathcal{D}$ is a contraction.
Proof
We prove for all $s \in Stat$ that $d\ (\Psi_\mathcal{D}\ (F)(s),\ \Psi_\mathcal{D}\ (G)(s)) \leq \frac{1}{2}\ d\ (F,\ G)$ using induction on the complexity of statement s.

1. Let $s \equiv (a,\ e)$ and $\epsilon = \mathcal{E}\ (e)$.
   $d\ (\Psi_\mathcal{D}\ (F)((a,\ e)),\ \Psi_\mathcal{D}\ (G)((a,\ e)))$
   $=$
   $d\ (\{\ (a,\ \epsilon)\ \},\ \{\ (a,\ \epsilon)\ \})$
   $=$
   0
   $\leq$
   $\frac{1}{2}\ d\ (F,\ G)$

2. Let $s \equiv x\ (e_1,\ \ldots,\ e_n)$ and $x\ (t_1,\ \ldots,\ t_n) \Leftarrow g \in d$.
   $d\ (\Psi_\mathcal{D}\ (F)(x\ (e_1,\ \ldots,\ e_n)),\ \Psi_\mathcal{D}\ (G)(x\ (e_1,\ \ldots,\ e_n)))$
   $=$
   $d\ (\Psi_\mathcal{D}\ (F)(g\ [e_1/t_1, \ldots e_n/t_n]),\ \Psi_\mathcal{D}\ (G)(g\ [e_1/t_1, \ldots e_n/t_n]))$
   $\leq$
   $\frac{1}{2}\ d\ (F,\ G)$

3. Let $s \equiv s_1\ ;\ s_2$.
   $d\ (\Psi_\mathcal{D}\ (F)(s_1\ ;\ s_2),\ \Psi_\mathcal{D}\ (G)(s_1\ ;\ s_2))$
   $=$
   $d\ (\Psi_\mathcal{D}\ (F)(s_1) \bullet F\ (s_2),\ \Psi_\mathcal{D}\ (G)(s_1) \bullet G\ (s_2))$
   $\leq$ # property 4.15 #
   $max\ (d\ (\Psi_\mathcal{D}\ (F)(s_1),\ \Psi_\mathcal{D}\ (G)(s_1)),\ \frac{1}{2}\ d\ (F\ (s_2),\ G\ (s_2)))$

$\leq$
$max\ (\frac{1}{2}\ d\ (F,\ G),\ \frac{1}{2}\ d\ (F,\ G))$
$=$
$\frac{1}{2}\ d\ (F,\ G)$

4. Let $s \equiv s_1 \cup s_2$.
   $d\ (\Psi_{\mathcal{D}}\ (F)(s_1 \cup s_2),\ \Psi_{\mathcal{D}}\ (G)(s_1 \cup s_2))$
   $=$
   $d\ (\Psi_{\mathcal{D}}\ (F)(s_1) \cup \Psi_{\mathcal{D}}\ (F)(s_2),\ \Psi_{\mathcal{D}}\ (G)(s_1) \cup \Psi_{\mathcal{D}}\ (G)(s_2))$
   $\leq$ # property A.11 #
   $max\ (d\ (\Psi_{\mathcal{D}}\ (F)(s_1),\ \Psi_{\mathcal{D}}\ (G)(s_1)),\ d\ (\Psi_{\mathcal{D}}\ (F)(s_2),\ \Psi_{\mathcal{D}}\ (G)(s_2)))$
   $\leq$
   $max\ (\frac{1}{2}\ d\ (F,\ G),\ \frac{1}{2}\ d\ (F,\ G))$
   $=$
   $\frac{1}{2}\ d\ (F,\ G)$

End 4.17

The mapping $\Psi_{\mathcal{D}}$ has a unique fixed point, which can be conclude from Banach's theorem. $\mathcal{D}_d$ is derived from $\Psi_{\mathcal{D}}$ using the fixed point property.

Finally we extend the denotational semantics from statements to programs in the last definition of this section.

Definition 4.18
The mapping $\mathcal{D} : Prog \rightarrow \mathcal{P}_{nc}\ (TS)$ is given by
$\mathcal{D}\ (d \mid s) = \mathcal{D}_d\ (s)$
End 4.18

5 Equivalence proof

Having defined both an operational and a denotational semantics for our language the question rises whether the denotational model $\mathcal{D}$ is correct with respect to the computational intuition captured by the operational model $\mathcal{O}$. We observe that the denotational model $\mathcal{D}$ does not record failures. Therefore we define a function $fail$ which introduces failures in timed streams. We will show that

$\mathcal{O} = fail\ (0) \circ \mathcal{D}$.

In order to prove this we define a so-called intermediate semantics $\mathcal{I}$. We define this intermediate semantics with the help of a labelled transition system like we have defined the operational semantics. This intermediate semantic model does not record failures just like the denotational semantics. We will prove that

$\mathcal{I} = \mathcal{D}$.

Finally, we relate the labelled transition systems defining the operational and intermediate semantics resulting in

$\mathcal{O} = fail\ (0) \circ \mathcal{I}$.

We first give the labelled transition system describing the intermediate model. This transition system is again associated with a declaration d. The set of configurations consists of $Stat_E$ where

$Stat_E = Stat \cup \{E\}$

and the set of labels consists of the set of timed actions. The rules associated with the declaration d are

$(a,\ e) -(a,\ \epsilon)\rightarrow_d E$ $\qquad\qquad \mathcal{E}\ (e) = \epsilon$

$$\frac{g\ [e_1/t_1,\ \ldots,\ e_n/t_n] -\alpha\rightarrow_d \bar{s}}{x\ (e_1,\ \ldots,\ e_n) -\alpha\rightarrow_d \bar{s}} \qquad x\ (t_1,\ \ldots,\ t_n) \Leftarrow g \in d$$

$$\frac{s -\alpha\rightarrow_d \bar{s}}{s\ ;\ s' -\alpha\rightarrow_d \bar{s}\ ;\ s'}$$

$$\frac{s -\alpha\rightarrow_d \bar{s}}{s \cup s' -\alpha\rightarrow_d \bar{s}} \qquad s' \cup s -\alpha\rightarrow_d \bar{s}$$

$$\frac{s -\alpha\rightarrow_d \bar{s}}{s \parallel s' -\alpha\rightarrow_d \bar{s} \parallel s'} \qquad s' \parallel s -\alpha\rightarrow_d s' \parallel \bar{s}$$

$$\frac{s\ [r/t] -\alpha\rightarrow_d \bar{s}}{\int_{t\in T}\ s -\alpha\rightarrow_d \bar{s}} \qquad r \in T$$

Using the above rules, we can derive $(a,\ 3)\ ;\ (b,\ 4) -(a,\ 3)\rightarrow_d (b,\ 4) -(b,\ 4)\rightarrow_d E$ and $(b,\ 4)\ ;\ (a,\ 3) -(b,\ 4)\rightarrow_d (a,\ 3) -(a,\ 3)\rightarrow_d E$.

Having defined the labelled transition system we can give the intermediate semantics for statements.

Definition 5.1
The mapping $\mathcal{I}_d : Stat \rightarrow \mathcal{P}_{nc}\ (TS)$ is given by
$\mathcal{I}_d\ (s) = \bigcup \{\ \alpha \cdot \mathcal{I}_d\ (s') \mid s -\alpha\rightarrow_d s'\ \} \cup \{\ \alpha \mid s -\alpha\rightarrow_d E\ \}$
End 5.1

The fact that the intermediate semantics $\mathcal{I}_d$ is well-defined can be proved along the lines of the well-definedness proof of the operational semantics $\mathcal{O}_d$. We only introduce the higher-order mapping $\Psi_{\mathcal{I}}$, which will be used in lemma 5.4.

Definition 5.2
The mapping $\Psi_{\mathcal{I}} : (Stat \rightarrow \mathcal{P}_{nc}\ (TS)) \rightarrow (Stat \rightarrow \mathcal{P}_{nc}\ (TS))$ is given by
$\Psi_{\mathcal{I}}\ (F)(s) = \bigcup \{\ \alpha \cdot F\ (s') \mid s -\alpha\rightarrow_d s'\ \} \cup \{\ \alpha \mid s -\alpha\rightarrow_d E\ \}$
End 5.2

Now we introduce the intermediate semantics for programs.

Definition 5.3
The mapping $\mathcal{I} : Prog \rightarrow \mathcal{P}_{nc}\ (TS)$ is given by
$\mathcal{I}\ (d \mid s) = \mathcal{I}_d\ (s)$
End 5.3

We have all the ingredients to prove $\mathcal{I} = \mathcal{D}$. First we show that $\Psi_{\mathcal{I}}\ (\mathcal{D}_d) = \mathcal{D}_d$. Using Banach's theorem we can conclude that $\mathcal{I}_d = \mathcal{D}_d$. From this we can easily deduce $\mathcal{I} = \mathcal{D}$.

Lemma 5.4
$\Psi_{\mathcal{I}}\ (\mathcal{D}_d) = \mathcal{D}_d$
Proof
We prove for all $s \in Stat$ that $\Psi_{\mathcal{I}}\ (\mathcal{D}_d)(s) = \mathcal{D}_d\ (s)$ using induction on the complexity of statement s.

1. Let $s \equiv (a,\ e)$ and $\mathcal{E}\ (e) = \epsilon$.
   $\Psi_{\mathcal{I}}\ (\mathcal{D}_d)((a,\ e))$
   $=$

$\bigcup \{ \alpha \cdot \mathcal{D}_d\ (s') \mid (a, e) -\alpha\rightarrow_d s' \} \cup \{ \alpha \mid (a, e) -\alpha\rightarrow_d E \}$
$=$
$\{(a, \epsilon)\}$
$=$
$\mathcal{D}_d\ ((a, e))$

2. Let $s \equiv x\ (e_1, \ldots, e_n)$ and $x\ (t_1, \ldots, t_n) \Leftarrow g$.
$\Psi_{\mathcal{I}}\ (\mathcal{D}_d)(x\ (e_1, \ldots, e_n))$
$=$
$\bigcup \{ \alpha \cdot \mathcal{D}_d\ (s') \mid x\ (e_1, \ldots, e_n) -\alpha\rightarrow_d s' \} \cup \{ \alpha \mid x\ (e_1, \ldots, e_n) -\alpha\rightarrow_d E \}$
$=$
$\bigcup \{ \alpha \cdot \mathcal{D}_d\ (s') \mid g\ [e_1/t_1, \ldots, e_n/t_n] -\alpha\rightarrow_d s' \} \cup \{ \alpha \mid g\ [e_1/t_1, \ldots, e_n/t_n] -\alpha\rightarrow_d E \}$
$=$
$\Psi_{\mathcal{I}}\ (\mathcal{D}_d)\ (g\ [e_1/t_1, \ldots, e_n/t_n])$
$=$
$\mathcal{D}_d\ (g\ [e_1/t_1, \ldots, e_n/t_n])$
$=$
$\mathcal{D}_d\ (x\ (e_1, \ldots, e_n))$

5. Let $s \equiv s_1 \parallel s_2$.

$\Psi_{\mathcal{I}}\ (\mathcal{D}_d)(s_1 \parallel s_2)$
$=$
$\bigcup \{ \alpha \cdot \mathcal{D}_d\ (s') \mid s_1 \parallel s_2 -\alpha\rightarrow_d s' \} \cup \{ \alpha \mid s_1 \parallel s_2 -\alpha\rightarrow_d E \}$
$=$
$\bigcup \{ \alpha \cdot \mathcal{D}_d\ (s' \parallel s_2) \mid s_1 -\alpha\rightarrow_d s' \} \cup \bigcup \{ \alpha \cdot \mathcal{D}_d\ (s_2) \mid s_1 -\alpha\rightarrow_d E \} \cup$
$\bigcup \{ \alpha \cdot \mathcal{D}_d\ (s_1 \parallel s') \mid s_2 -\alpha\rightarrow_d s' \} \cup \bigcup \{ \alpha \cdot \mathcal{D}_d\ (s_1) \mid s_2 -\alpha\rightarrow_d E \}$
$=$
$\bigcup \{ \alpha \cdot (\mathcal{D}_d\ (s') \parallel \mathcal{D}_d\ (s_2)) \mid s_1 -\alpha\rightarrow_d s' \} \cup \bigcup \{ \alpha \cdot \mathcal{D}_d\ (s_2) \mid s_1 -\alpha\rightarrow_d E \} \cup$
$\bigcup \{ \alpha \cdot (\mathcal{D}_d\ (s_1) \parallel \mathcal{D}_d\ (s')) \mid s_2 -\alpha\rightarrow_d s' \} \cup \bigcup \{ \alpha \cdot \mathcal{D}_d\ (s_1) \mid s_2 -\alpha\rightarrow_d E \}$
$=$
$\bigcup \{ \alpha \cdot (\sigma \parallel \tau) \mid \sigma \in \mathcal{D}_d\ (s') \wedge \tau \in \mathcal{D}_d\ (s_2) \wedge s_1 -\alpha\rightarrow_d s' \} \cup$
$\{ \alpha \cdot \tau \mid \tau \in \mathcal{D}_d\ (s_2) \wedge s_1 -\alpha\rightarrow_d E \} \cup$
$\bigcup \{ \alpha \cdot (\sigma \parallel \tau) \mid \sigma \in \mathcal{D}_d\ (s_1) \wedge \tau \in \mathcal{D}_d\ (s') \wedge s_2 -\alpha\rightarrow_d s' \} \cup$
$\{ \alpha \cdot \sigma \mid \sigma \in \mathcal{D}_d\ (s_1) \wedge s_2 -\alpha\rightarrow_d E \}$
$=$
$\bigcup \{ (\alpha \cdot \sigma) \between \tau \mid \sigma \in \mathcal{D}_d\ (s') \wedge \tau \in \mathcal{D}_d\ (s_2) \wedge s_1 -\alpha\rightarrow_d s' \} \cup$
$\{ \alpha \between \tau \mid \tau \in \mathcal{D}_d\ (s_2) \wedge s_1 -\alpha\rightarrow_d E \} \cup$
$\bigcup \{ (\alpha \cdot \tau) \between \sigma \mid \sigma \in \mathcal{D}_d\ (s_1) \wedge \tau \in \mathcal{D}_d\ (s') \wedge s_2 -\alpha\rightarrow_d s' \} \cup$
$\{ \alpha \between \sigma \mid \sigma \in \mathcal{D}_d\ (s_1) \wedge s_2 -\alpha\rightarrow_d E \}$
$=$
$(\bigcup \{ (\alpha \cdot \mathcal{D}_d\ (s') \mid s_1 -\alpha\rightarrow_d s' \} \cup \{ \alpha \mid s_1 -\alpha\rightarrow_d E \}) \between \mathcal{D}_d\ (s_2) \cup$
$(\bigcup \{ (\alpha \cdot \mathcal{D}_d\ (s') \mid s_2 -\alpha\rightarrow_d s' \} \cup \{ \alpha \mid s_2 -\alpha\rightarrow_d E \}) \between \mathcal{D}_d\ (s_1)$
$=$
$\Psi_{\mathcal{I}}\ (\mathcal{D}_d)(s_1) \between \mathcal{D}_d\ (s_2) \cup \Psi_{\mathcal{I}}\ (\mathcal{D}_d)(s_2) \between \mathcal{D}_d\ (s_1)$
$=$
$\mathcal{D}_d\ (s_1) \between \mathcal{D}_d\ (s_2) \cup \mathcal{D}_d\ (s_2) \between \mathcal{D}_d\ (s_1)$
$=$
$\mathcal{D}_d\ (s_1) \parallel \mathcal{D}_d\ (s_2)$
$=$
$\mathcal{D}_d\ (s_1 \parallel s_2)$

6. Let $s \equiv \int_{t \in T} s$.
   $\Psi_{\mathcal{I}}\ (\mathcal{D}_d)(\int_{t \in T}\ s)$
   $=$
   $\bigcup \{\ \alpha \cdot \mathcal{D}_d\ (s') \mid \int_{t \in T}\ s \mathrel{-}\alpha\mathrel{\rightarrow_d}\ s'\ \} \cup \{\ \alpha \mid \int_{t \in T}\ s \mathrel{-}\alpha\mathrel{\rightarrow_d}\ E\ \}$
   $=$
   $\bigcup \{\ \bigcup \{\ \alpha \cdot \mathcal{D}_d\ (s') \mid s\ [r/t] \mathrel{-}\alpha\mathrel{\rightarrow_d}\ s'\ \} \mid r \in T\ \} \cup \bigcup \{\ \{\ \alpha \mid s\ [r/t] \mathrel{-}\alpha\mathrel{\rightarrow_d}\ E\ \} \mid r \in T\ \}$
   $=$
   $\bigcup \{\ \bigcup \{\ \alpha \cdot \mathcal{D}_d\ (s') \mid s\ [r/t] \mathrel{-}\alpha\mathrel{\rightarrow_d}\ s'\ \} \cup \{\ \alpha \mid s\ [r/t] \mathrel{-}\alpha\mathrel{\rightarrow_d}\ E\ \} \mid r \in T\ \}$
   $=$
   $\bigcup \{\ \Psi_{\mathcal{I}}\ (s\ [r/t]) \mid r \in T\ \}$
   $=$
   $\bigcup \{\ \mathcal{D}_d\ (s\ [r/t]) \mid r \in T\ \}$
   $=$
   $\mathcal{D}_d\ (\int_{t \in T}\ s)$

End 5.4

We have left to prove that $\mathcal{O} = fail\ (0) \circ \mathcal{I}$. First we give the function $fail$ which introduces failures in timed streams. The function $fail\ (r)$ identifies those streams σ whose first action occurs after time r and whose actions are in the right order, i.e. if $(a,\ r)$ precedes $(a',\ r')$ in σ then $r < r'$. If the first action of a stream σ occurs at or before r, $fail\ (r)(\sigma)$ delivers $(\delta,\ r)$. If the actions of stream σ are not in the right order, $fail\ (r)(\sigma)$ gives the longest prefix of σ, which is in the right order, concatenated with a failure after the time stamp of the last action of this longest prefix.

Definition 5.5
The mapping $fail : I\!R_{\geq} \rightarrow TS \rightarrow TFS$ is given by

$fail\ (r)((a,\ r')) = (a,\ r')$	$r < r'$
$fail\ (r)((a,\ r')) = (\delta,\ r)$	$r \geq r'$
$fail\ (r)((a,\ r') \cdot \sigma) = (a,\ r') \cdot fail\ (r')(\sigma)$	$r < r'$
$fail\ (r)((a,\ r') \cdot \sigma) = (\delta,\ r)$	$r \geq r'$

The mapping $fail : I\!R_{\geq} \rightarrow \mathcal{P}_{nc}\ (TS) \rightarrow \mathcal{P}_{nc}\ (TFS)$ is given by
$fail\ (r)(\Sigma) = \{\ fail\ (r)(\sigma) \mid \sigma \in \Sigma\ \}$
End 5.5

For example, we have that $fail\ (0)((a,\ 3)(b,\ 4)) = (a,\ 3)\ fail\ (3)((b,\ 4)) = (a,\ 3)(b,\ 4)$ and $fail\ (0)((b,\ 4)(a,\ 3)) = (b,\ 4)\ fail\ (4)((a,\ 3)) = (b,\ 4)(\delta,\ 4)$.

Proving that $fail$ is well-defined is left to the reader. The proof follows from the fact that if $fail\ (r)(\sigma)$ is infinite, then we have that $fail\ (r)(\sigma) = \sigma$.

Next we prove two properties which relate the transition systems describing the operational and intermediate semantics.

Property 5.6
For all $s \in Stat$, $\bar{s} \in Stat_E$, $a \in Atom$ and $r,\ r' \in I\!R_{\geq}$
$s \mathrel{-}(a,\ r')\mathrel{\rightarrow_d}\ \bar{s} \wedge r < r' \Leftrightarrow [s,\ r] \mathrel{-}(a,\ r')\mathrel{\rightarrow_d}\ [\bar{s},\ r']$
Proof
We prove this using induction on the complexity of statement s.

1. Let $s \equiv (a',\ e)$ and $\epsilon = \mathcal{E}\ (e)$. By inspection of the transition systems we can write the left-hand side of the equivalence as $(a',\ e) \mathrel{-}(a',\ \epsilon)\mathrel{\rightarrow_d}\ E \wedge r < \epsilon$ and the right-hand side as $[(a',\ e),\ r] \mathrel{-}(a',\ \epsilon)\mathrel{\rightarrow_d}\ [E,\ \epsilon]$, where $r < \epsilon$. We can conclude from this that the property is satisfied in this case.

2. Let $s \equiv s_1 \ ; \ s_2$.
   $s_1 \ ; \ s_2 \ -(a, \ r')\rightarrow_d \ \bar{s} \wedge r < r'$
   $\Leftrightarrow$
   $s_1 \ -(a, \ r')\rightarrow_d \ \bar{s}' \wedge r < r' \wedge \bar{s} \equiv \bar{s}' \ ; \ s_2$
   $\Leftrightarrow$
   $[s_1, \ r] \ -(a, \ r')\rightarrow_d \ [\bar{s}', \ r'] \wedge \bar{s} \equiv \bar{s}' \ ; \ s_2$
   $\Leftrightarrow$
   $[s_1 \ ; \ s_2, \ r] \ -(a, \ r')\rightarrow_d \ [\bar{s}, \ r']$

4. Let $s \equiv s_1 \cup s_2$.
   $s_1 \cup s_2 \ -(a, \ r')\rightarrow_d \ \bar{s} \wedge r < r'$
   $\Leftrightarrow$
   $(s_1 \ -(a, \ r')\rightarrow_d \ \bar{s} \wedge r < r') \vee (s_2 \ -(a, \ r')\rightarrow_d \ \bar{s} \wedge r < r')$
   $\Leftrightarrow$
   $[s_1, \ r] \ -(a, \ r')\rightarrow_d \ [\bar{s}, \ r'] \vee [s_2, \ r] \ -(a, \ r')\rightarrow_d \ [\bar{s}, \ r']$
   $\Leftrightarrow$
   $[s_1 \cup s_2, \ r] \ -(a, \ r')\rightarrow_d \ [\bar{s}, \ r']$

End 5.6

From this property we can deduce that the empty statement E of the operational semantics is related to the empty statement of the intermediate statement.

Property 5.7
For all $s \in Stat$, $\bar{s} \in Stat_E$, $a \in Atom$ and $r, \ r' \in I\!R_{\geq}$
$s \ -(a, \ r')\rightarrow_d \ \bar{s} \wedge r \geq r' \Leftrightarrow [s, \ r] \ -(\delta, \ r)\rightarrow_d \ [\Delta, \ r]$
Proof
We prove this using induction on the complexity of statement s.

1. Let $s \equiv (a', \ e)$ and $\epsilon = \mathcal{E} \ (e)$. By inspection of the transition systems we can write the left-hand side of the equivalence as $(a', \ e) \ -(a', \ \epsilon)\rightarrow_d \ E \wedge r \geq \epsilon$ and the right-hand side as $[(a', \ e), \ r] \ -(\delta, \ r)\rightarrow_d \ [\Delta, \ r]$, where $r \geq \epsilon$. We can conclude from this that the property is satisfied in this case.

2. Let $s \equiv x \ (e_1, \ \ldots, \ e_n)$ and $x \ (t_1, \ \ldots, \ t_n) \Leftarrow g \in d$.
   $x \ (e_1, \ \ldots, \ e_n) \ -(a, \ r')\rightarrow_d \ \bar{s} \wedge r \geq r'$
   $\Leftrightarrow$
   $g \ [e_1/t_1, \ \ldots, \ e_n/t_n] \ -(a, \ r')\rightarrow_d \ \bar{s} \wedge r \geq r'$
   $\Leftrightarrow$
   $[g \ [e_1/t_1, \ \ldots, \ e_n/t_n] \ , \ r] \ -(\delta, \ r)\rightarrow_d \ [\Delta, \ r]$
   $\Leftrightarrow$
   $[x \ (e_1, \ \ldots, \ e_n), \ r] \ -(\delta, \ r)\rightarrow_d \ [\Delta, \ r]$

5. Let $s \equiv s_1 \parallel s_2$.
   $s_1 \parallel s_2 \ -(a, \ r')\rightarrow_d \ \bar{s} \wedge r \geq r'$
   $\Leftrightarrow$
   $(s_1 \ -(a, \ r')\rightarrow_d \ \bar{s}' \wedge r \geq r' \wedge \bar{s} \equiv \bar{s}' \parallel s_2) \vee (s_2 \ -(a, \ r')\rightarrow_d \ \bar{s}' \wedge r \geq r' \wedge \bar{s} \equiv s_1 \parallel \bar{s}')$
   $\Leftrightarrow$
   $([s_1, \ r] \ -(\delta, \ r)\rightarrow_d \ [\Delta, \ r] \wedge \bar{s} \equiv \bar{s}' \parallel s_2) \vee ([s_2, \ r] \ -(\delta, \ r)\rightarrow_d \ [\Delta, \ r] \wedge \bar{s} \equiv s_1 \parallel \bar{s}')$
   $\Leftrightarrow$
   $[s_1 \parallel s_2, \ r] \ -(\delta, \ r)\rightarrow_d \ [\Delta, \ r]$

6. Let $s \equiv \int_{t \in T} s$.
   $\int_{t \in T} \ s \ -(a, \ r')\rightarrow_d \ \bar{s} \wedge r \geq r'$

$\Leftrightarrow$
$\exists\ r'' \in T\ s\ [r''/t] -(a,\ r')\rightarrow_d \bar{s} \wedge r \geq r'$
$\Leftrightarrow$
$\exists\ r'' \in T\ [s\ [r''/t],\ r] -(\delta,\ r)\rightarrow_d [\Delta,\ r]$
$\Leftrightarrow$
$[\int_{t \in T}\ s,\ r] -(\delta,\ r)\rightarrow_d [\Delta,\ r]$

End 5.7

The above property tells us that the failure statement Δ is related to an arbitrary statement of the intermediate semantics.

We introduce the mapping $\mathcal{I}_d^{fail}$ in order to prove that $\mathcal{O}_d\ ([s,\ r]) = fail\ (r)(\mathcal{I}_d\ (s))$.

Definition 5.8
The mapping $\mathcal{I}_d^{fail} : Conf \rightarrow \mathcal{P}_{nc}\ (TFS)$ is given by
$\mathcal{I}_d^{fail}\ ([s,\ r]) = fail\ (r)(\mathcal{I}_d\ (s))$
End 5.8

We show that $\Psi_{\mathcal{O}}\ (\mathcal{I}_d^{fail}) = \mathcal{I}_d^{fail}$. Again using Banach's theorem we can conclude that $\mathcal{O}_d = \mathcal{I}_d^{fail}$, which gives us $\mathcal{O}_d\ ([s,\ r]) = fail\ (r)(\mathcal{I}_d\ (s))$.

Lemma 5.9
$\Psi_{\mathcal{O}}\ (\mathcal{I}_d^{fail}) = \mathcal{I}_d^{fail}$
Proof
We prove for all $C \in Conf$ that $\Psi_{\mathcal{O}}\ (\mathcal{I}_d^{fail})(C) = \mathcal{I}_d^{fail}\ (C)$. Let $C \equiv [s,\ r]$.
$\Psi_{\mathcal{O}}\ (\mathcal{I}_d^{fail})([s,\ r])$
$=$
$\bigcup \{\ l \cdot \mathcal{I}_d^{fail}\ (C') \mid [s,\ r] -l\rightarrow_d C'\ \} \cup \{\ l \mid [s,\ r] -l\rightarrow_d [E,\ r']\ \} \cup \{\ l \mid [s,\ r] -l\rightarrow_d [\Delta,\ r']\ \}$
$=$
$\bigcup \{\ (a,\ r) \cdot \mathcal{I}_d^{fail}\ ([s',\ r']) \mid [s,\ r] -(a,\ r)\rightarrow_d [s',\ r']\ \} \cup$
$\quad \{\ (a,\ r) \mid [s,\ r] -(a,\ r)\rightarrow_d [E,\ r']\ \} \cup$
$\quad \{\ (a,\ r) \mid [s,\ r] -(a,\ r)\rightarrow_d [\Delta,\ r']\ \}$
$=$ # property 5.6 and property 5.7 #
$\bigcup \{\ (a,\ r) \cdot \mathcal{I}_d^{fail}\ ([s',\ r']) \mid s -(a,\ r)\rightarrow_d s' \wedge r < r'\ \} \cup$
$\quad \{\ (a,\ r) \mid s -(a,\ r)\rightarrow_d E \wedge r < r'\ \} \cup$
$\quad \{\ (\delta,\ r) \mid s -(a,\ r)\rightarrow_d s' \wedge r \geq r'\ \} \cup$
$\quad \{\ (\delta,\ r) \mid s -(a,\ r)\rightarrow_d E \wedge r \geq r'\ \}$
$=$
$\bigcup \{\ (a,\ r) \cdot fail\ (r')(\mathcal{I}_d\ (s')) \mid s -(a,\ r)\rightarrow_d s' \wedge r < r'\ \} \cup$
$\quad \{\ (a,\ r) \mid s -(a,\ r)\rightarrow_d E \wedge r < r'\ \} \cup$
$\quad \{\ (\delta,\ r) \mid s -(a,\ r)\rightarrow_d s' \wedge r \geq r'\ \} \cup$
$\quad \{\ (\delta,\ r) \mid s -(a,\ r)\rightarrow_d E \wedge r \geq r'\ \}$
$=$
$\bigcup \{\ fail\ (r)(\alpha \cdot \mathcal{I}_d\ (s')) \mid s -\alpha\rightarrow_d s'\ \} \cup \{\ fail\ (r)(\alpha) \mid s -\alpha\rightarrow_d E\ \}$
$=$
$fail\ (r)(\bigcup \{\ \alpha \cdot \mathcal{I}_d\ (s') \mid s -\alpha\rightarrow_d s'\ \} \cup \{\ \alpha \mid s -\alpha\rightarrow_d E\ \})$
$=$
$fail\ (r)(\mathcal{I}_d\ (s))$
$=$
$\mathcal{I}_d^{fail}\ ([s,\ r])$
End 5.9

We conclude the equivalence proof and this section by showing that $\mathcal{O} = fail\ (0) \circ \mathcal{I}$.

Theorem 5.10
$\mathcal{O} = fail\ (0) \circ \mathcal{I}$
Proof
We prove for all $p \in Prog$ that $\mathcal{O}\ (p) = (fail\ (0) \circ \mathcal{I})\ (p)$. Let $p \equiv d \mid s$.
$\mathcal{O}\ (d \mid s)$
=
$\mathcal{O}'_d\ ([s, 0])$
= # lemma 3.10 #
$\mathcal{O}_d\ ([s, 0])$
= # lemma 5.9 #
$fail\ (0)(\mathcal{I}_d\ (s))$
=
$fail\ (0)(\mathcal{I}\ (d \mid s))$
=
$(fail\ (0) \circ \mathcal{I})\ (d \mid s))$
End 5.10

6 Concluding remarks

We can conclude that the metric approach for defining models can also be applied to languages incorporating time related aspects like timed atomic actions and integration.

It seems possible to give another denotational model $\mathcal{D}$ leading to the equivalence result $\mathcal{O} = \mathcal{D}$. We expect it to be possible to relate the denotational semantics defined in [RR86, RR87, R88] to a denotational semantics based on the denotational model presented in this paper following the lines of [BMO87]. Finding other criteria to restrict time sets and enriching the language with communication and global non-determinism are still issues for further research.

Acknowledgements

We would like to thank the members of the Amsterdam Concurrency Group for comments on a previous version of this paper. In particular, we thank Erik de Vink for his comments and suggestions during the evolvement of this paper.

References

[A81] K.R. Apt. *Recursive Assertions and Parallel Programs.* Acta Informatica 15 (1981), 219-232.

[AB88] P. America and J.W. de Bakker. *Designing equivalent semantic models for process creation.* in: Proceedings Advanced School on Mathematical models for the Semantic Parallelism (M. Venturini Zilli, ed.), Lecture Notes in Computer Science 280, Springer (1988), 109-176.

[ABKR86] P. America, J.W. de Bakker, J.N. Kok and J.J.M.M. Rutten. *Denotational semantics for a parallel object-oriented language.* Information and Computation 83 (1989), 152-205.

[B88] J.W. de Bakker. *Comparative semantics for flow of control in logic programming without logic.* Report CS-8840, Centre for Mathematics and Computer Science, Amsterdam (1988). To appear in Information and Computation.

[BB89] J.C.M. Baeten and J.A. Bergstra. *Real Time Process Algebra.* Report P8916, Programming Research Group, University of Amsterdam (1989).

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

[BC85] G. Berry and L. Cosserat. *The ESTEREL Synchronous Programming Language and its Semantics.* in: Proceedings CMU Seminar on Concurrency (S.D. Brookes, A.W. Roscoe and G. Winksel, eds.), Lecture Notes in Computer Science 197, Springer (1985), 389-448.

[BK82] J.A. Bergstra and J.W. Klop. *Fixed Point Semantics in Process Algebra.* Report IW 206/82, Mathematical Centre, Amsterdam (1982).

[BK88] J.W. de Bakker and J.N. Kok. *Uniform abstraction, atomicity and contractions in the comparative semantics of Concurrent Prolog.* in: Proceedings International Conference on Fifth Generation Computer Systems 1988, Institute for New Generation Computer Technology (1988), 347-355.

[BM88] J.W. de Bakker and J.-J.Ch. Meyer. *Metric semantics for concurrency.* BIT 28 (1988), 504-529.

[BMO87] J.W. de Bakker, J.-J.Ch. Meyer and E.-R. Olderog. *Infinite streams and finite observations in the semantics of uniform concurrency.* Theoretical Computer Science 49 (1987), 87-112.

[BKMOZ86] J.W. de Bakker, J.N. Kok, J.-J.Ch. Meyer, E.-R. Olderog and I.J. Zucker. *Contrasting themes in the semantics of imperative concurrency.* in: Current Trends in Concurrency: Overviews and Tutorials (J.W. de Bakker, W.P. de Roever and G. Rozenberg, eds.), Lecture Notes in Computer Science 224, Springer (1986), 51-121.

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

[DS89] J.W. Davies and S.A. Schneider. *An introduction to timed CSP.* Report PRG-75, Oxford University Computing Laboratory, Oxford (1989).

[E89] R. Engelking. *General Topology.* Revised and completed version. Sigma Series in Pure Mathematics 6, Heldermann Verlag Berlin (1989).

[GV89] J.F. Groote and F.W. Vaandrager. *Structured Operational Semantics and Bisimulation as a Congruence.* in: Proceedings 16th International Colloquium on Automata, Languages and Programming (G. Ausiello, M. Dezani-Ciancaglini and S. Ronchi Della Rocca, eds.), Lecture Notes in Computer Science 372, Springer (1989), 423-438.

[HP79] M. Hennessy and G.D. Plotkin. *Full abstraction for a simple parallel programming language.* in: Proceedings 8th Mathematical Foundations of Computer Science (J. Becvar, ed.), Lecture Notes in Computer Science 74, Springer (1979), 108-120.

[KR88] J.N. Kok and J.J.M.M. Rutten. *Contractions in Comparing Semantics.* in: Proceedings 15th International Colloquium on Automata, Languages and Programming (T. Lepistö, A. Salomaa, eds.), Lecture Notes in Computer Science 317, Springer (1988), 317-332.

[LZ88] I. Lee and A. Zwarico. *Times Acceptances: A Model of Time Dependent Processes.* in: Proceedings Formal Techniques in Real-Time and Fault-Tolerant Systems (M. Joseph, ed.), Lecture Notes in Computer Science 331, Springer (1988), 128-130.

[R88] G.M. Reed. *A Uniform Mathematical Theory of Real-time Distributed Computing.* Ph.D. thesis, Oxford University (1988).

[RR86] G.M. Reed and A.W. Roscoe. *A Timed Model for Communicating Sequential Processes.* in: Proceedings 13th International Colloquium on Automata, Languages and Programming (L. Kott, ed.), Lecture Notes in Computer Science 226, Springer (1986), 314-323. Theoretical Computer Science 58 (1988), 249-261.

[RR87] G.M. Reed and A.W. Roscoe. *Metric spaces as models for real-time concurrency.* in: Proceedings Mathematical Foundations of Programming Languages and Semantics (M. Main, A. Melton, M. Mislove and D. Schmidt, eds.), Lecture Notes in Computer Science 298, Springer (1987), 331-343.

A Mathematical preliminaries

In this appendix we introduce the notions metric space [E89] and labelled transition system.

Definition A.1
A metric space is a tuple (X, d) where X is a non-empty set and d is a mapping
$d : X \times X \rightarrow [0, 1]$, which we call metric or distance, that satisfies the following properties.

- $\forall\ x \in X\ \forall\ y \in X \quad d\ (x, y) = 0 \Leftrightarrow x = y$
- $\forall\ x \in X\ \forall\ y \in X \quad d\ (x, y) = d\ (y, x)$
- $\forall\ x \in X\ \forall\ y \in X\ \forall\ z \in X \quad d\ (x, z) \leq d\ (x, y) + d\ (y, z)$

End A.1

In the sequel, we will define some metric spaces on finite and infinite streams over some set A. With $A^\infty = A^* \cup A^\omega$ we denote the set of all finite and infinite streams over A. For $\sigma \in A^\infty$ and $n \in I\!N$ $\sigma\ [n]$ denotes the prefix of length n of stream σ, in the case the length of σ is greater than or equal to n, and σ otherwise. We put $d\ (\sigma, \tau) = 2^{-sup\{n | \sigma[n] = \tau[n]\}}$. The tuple (A^∞, d) is a metric space.

Definition A.2
Let (X, d) be a metric space and let $\{x_i\}_i$ be a sequence in X. We say that $\{x_i\}_i$ is a Cauchy sequence whenever we have that
$\forall\ \varepsilon > 0\ \exists\ N \in I\!N\ \forall\ n > N\ \forall\ m > N \quad d\ (x_n, x_m) < \varepsilon$
End A.2

For example, the sequence $\{a^i\}_i$ in A^∞ is a Cauchy sequence.

Definition A.3
Let (X, d) be a metric space ,$\{x_i\}_i$ be a sequence in X and $x \in X$. We say that $\{x_i\}_i$ converges to x and call x the limit of $\{x_i\}_i$ whenever we have that
$\forall\ \varepsilon > 0\ \exists\ N \in I\!N_>\ \forall\ n > N \quad d\ (x, x_n) < \varepsilon$
End A.3

For example, the sequence $\{a^i b\}_i$ in A^∞ converges to a^ω.

Definition A.4
A metric space (X, d) is called complete whenever each Cauchy sequence in X converges to an element in X.
End A.4

The metric space (A^∞, d) is complete. However, the metric space (A^*, d) is not complete.

Definition A.5
Let (X, d) be a metric space. A subset Y of X is called closed whenever each Cauchy sequence in Y converges to an element in Y.
End A.5

The set $\{a^i b\} \cup \{a^\omega\}$ is closed and the set $\{a^i b\}$ is not closed.

Definition A.6
Let (X_1, d_1) and (X_2, d_2) be metric spaces. We define a metric d on functions $f_1, f_2 \in X_1 \to X_2$ by
$d\ (f_1,\ f_2) = sup\ \{\ d_2\ (f_1\ (x),\ f_2\ (x)) \mid x \in X_1\ \}$
End A.6

Note that the above definition does not depend on the metric d_1 on X_1.

Definition A.7
Let (X_1, d_1) and (X_2, d_2) be metric spaces. Let $\varepsilon \geq 0$. With $X_1 \to^\varepsilon X_2$ we denote the set of functions f from X_1 to X_2 that satisfy
$\forall\ x \in X_1\ \forall\ y \in X_2 \quad d_2\ (f\ (x),\ f\ (y)) \leq \varepsilon \times d_1\ (x,\ y)$
The functions in $X_1 \to^1 X_2$ are called non-distance increasing and the functions in $X_1 \to^\varepsilon X_2$ with $0 \leq \varepsilon < 1$ are called contracting.
End A.7

We can extend a metric on sets to a metric on non-empty closed subsets of those sets as stated in the following definition.

Definition A.8
Let (X, d) be a metric space. We define a mapping d_H, the Hausdorff distance, on $\mathcal{P}_{nc}\ (X)$, the set of all non-empty closed subsets of X, by
$d_H\ (X, Y) = max\ \{\ sup\ x \in X\ inf\ y \in Y\ d\ (x,\ y),\ sup\ y \in Y\ inf\ x \in X\ d\ (y,\ x)\ \}$
End A.8

Next we state Hahn's theorem.

Theorem A.9
If (X, d) is a complete metric space, then $(\mathcal{P}_{nc}\ (X),\ d_H)$ is also a complete metric space.
End A.9

From this theorem we can derive that $(\mathcal{P}_{nc}\ (A^\infty),\ d_H)$ is a complete metric space.

Banach's theorem, which will often be used in the various sections of this paper, is stated in the following theorem.

Theorem A.10
If (X, d) is a complete metric space and $f : X \to X$ is a contraction then f has a unique fixed point x. Furthermore, we have that for all $y \in X\ lim_n\ f^n\ (y) = x$.
End A.10

The next property states that the operator $\bigcup$ is non-distance increasing in all its operands. This property will be useful to prove several contraction properties.

Property A.11
For all Σ_i, $\mathcal{T}_i \in \mathcal{P}_{nc}(A^\infty)$ and $\varepsilon \in [0, 1]$
$\forall\ i \quad d(\Sigma_i, \mathcal{T}_i) \leq \varepsilon \Rightarrow d(\bigcup \Sigma_i, \bigcup \mathcal{T}_i) \leq \varepsilon$
Proof
$\forall\ i \quad d(\Sigma_i, \mathcal{T}_i) \leq \varepsilon$
$\Leftrightarrow$
$\forall\ i \quad max\ \{\ sup\ \sigma \in \Sigma_i\ inf\ \tau \in \mathcal{T}_i \quad d(\sigma, \tau),\ sup\ \tau \in \mathcal{T}_i\ inf\ \sigma \in \Sigma_i \quad d(\tau, \sigma)\ \} \leq \varepsilon$
$\Leftrightarrow$
$\forall\ i \quad sup\ \sigma \in \Sigma_i\ inf\ \tau \in \mathcal{T}_i \quad d(\sigma, \tau) \leq \varepsilon \wedge \forall\ i \quad sup\ \tau \in \mathcal{T}_i\ inf\ \sigma \in \Sigma_i \quad d(\tau, \sigma) \leq \varepsilon$
$\Rightarrow$
$\forall\ i \quad sup\ \sigma \in \Sigma_i\ inf\ \tau \in \bigcup \mathcal{T}_i \quad d(\sigma, \tau) \leq \varepsilon \wedge \forall\ i \quad sup\ \tau \in \mathcal{T}_i\ inf\ \sigma \in \bigcup \Sigma_i \quad d(\tau, \sigma) \leq \varepsilon$
$\Rightarrow$
$sup\ \sigma \in \bigcup \Sigma_i\ inf\ \tau \in \bigcup \mathcal{T}_i \quad d(\sigma, \tau) \leq \varepsilon \wedge sup\ \tau \in \bigcup \mathcal{T}_i\ inf\ \sigma \in \bigcup \Sigma_i \quad d(\tau, \sigma) \leq \varepsilon$
$\Leftrightarrow$
$max\ \{\ sup\ \sigma \in \bigcup \Sigma_i\ inf\ \tau \in \bigcup \mathcal{T}_i \quad d(\sigma, \tau),\ sup\ \tau \in \bigcup \mathcal{T}_i\ inf\ \sigma \in \bigcup \Sigma_i \quad d(\tau, \sigma)\ \} \leq \varepsilon$
$\Leftrightarrow$
$d(\bigcup \Sigma_i, \bigcup \mathcal{T}_i) \leq \varepsilon$
End A.11

We conclude this section with the definition of a labelled transition system.

Definition A.12
A labelled transition system is a triple $(Conf, Label, \longrightarrow)$ consisting of a set of configurations $Conf$, a set of labels $Label$ and a transition relation $\longrightarrow \subseteq Conf \times Label \times Conf$.
End A.12

It is convenient to write $C -l\rightarrow_d C'$ instead of $(C, l, C') \in \longrightarrow$. When we defined a labelled transition system in one of the previous sections, we gave the set of configurations, the set of labels and the axioms and rules of a transition system specification. From this transition system specification, the transition relation of the labelled transition system can be derived as is described by Groote and Vaandrager in [GV89].