TU/e EINDHOVEN UNIVERSITY OF TECHNOLOGY

Discrete time process algebra

Citation for published version (APA):
Baeten, J. C. M., & Bergstra, J. A. (1992). *Discrete time process algebra*. (Reports of the programming research group, University of Amsterdam = Rapporten van de vakgroep programmatuur, Universiteit van Amsterdam; Vol. P9208). Universiteit van Amsterdam.

Document status and date:
Published: 01/01/1992

Document Version:
Publisher's PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:

• A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
• The final author version and the galley proof are versions of the publication after peer review.
• The final published version features the final layout of the paper including the volume, issue and page numbers.

Link to publication

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain
• You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the "Taverne" license above, please follow below link for the End User Agreement:

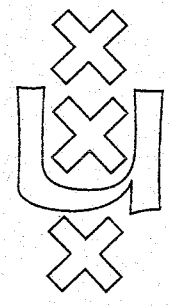
www.tue.nl/taverne

Take down policy
If you believe that this document breaches copyright please contact us at:

openaccess@tue.nl

providing details and we will investigate your claim.

Download date: 04. Oct. 2023



University of Amsterdam
Department of Mathematics and Computer Science
Programming Research Group

Discrete time process algebra

J.C.M. Baeten
J.A. Bergstra

Report P9208

April 1992

J.C.M. Baeten

Department of Computer Science
Eindhoven University of Technology

P.O. Box 513
5600 MB Eindhoven
The Netherlands

J.A. Bergstra

Programming Research Group
Department of Mathematics and Computer Science
University of Amsterdam

Kruislaan 403
1098 SJ Amsterdam
The Netherlands

tel. +31 20 5257591

Department of Philosophy
University at Utrecht

Heidelberglaan 8

P.O. Box 80106
3508 TC Utrecht

The Netherlands

Universiteit van Amsterdam, 1992

Discrete Time Process Algebra

J.C.M. Baeten
Department of Computing Science, Eindhoven University of Technology, P.O.Box 513, 5600 MB Eindhoven, The Netherlands

J.A. Bergstra
Programming Research Group, University of Amsterdam, Kruislaan 403, 1098 SJ Amsterdam, The Netherlands
and
Department of Philosophy, Utrecht University, Heidelberglaan 8, 3584 CS Utrecht, The Netherlands

The axiom system ACP of [BEK84] is extended to ACP_{dt}, which involves discrete time delay, and then to ACP_{dt} + ATP, an axiomatisation that adds key features of ATP [NIS90] to ACP. We give an interpretation of all discrete time constructs in the real time theory $ACP_{\rho\sqrt{I}}$.

1980 Mathematics Subject Classification (1985 revision): 68Q55, 68Q45, 68Q10.
1987 CR Categories: F.1.2, F.3.1, D.1.3, D.3.1.
Key words & Phrases: process algebra, real time process algebra, discrete time, time delay.
Note: Partial support received from ESPRIT Basic Research Action 3006, CONCUR. The second author also received partial support from RACE contract 1046, SPECS. This document does not necessarily reflect the views of the SPECS consortium.

1. INTRODUCTION.

Process algebra in the form of ACP [BEK84, BAW90], CCS [MIL89] or CSP [BRHR84] describes the main features of imperative concurrent programming without any explicit mention of time. Implicitly, time is present in the interpretation of sequential composition: in $p \cdot q$ (ACP notation) the process p must be executed before q. It may be felt as a weakness of the mentioned process algebra frameworks that time features implicitly only. Indeed a quantative view on the relation between process execution and progress of time is definitely absent in these calculi. This seems worse than it actually is. Nothing prevents one to introduce timer processes that represent clocks, to have processes interacting with these clocks and to have synchronisations between the clocks in ACP, CCS or CSP.

Of course timing aspects can be described more uniformly if process algebras are given that provide standardised features to incorporate a quantative view on time. Obvious as this may seem, the more obvious observation is that in a few years so many different formats for timed process algebras have been introduced that this uniformity seems to be further away than before. There are several reasons for the emerging spectrum of timed process algebras. We explain this by considering various options for adding time. The most obvious option is to represent time by means of non-negative reals, and to have time stamps on actions. This is done in [BAB91a] for ACP, in [RER88] for CSP and in [MOT90] for CCS. The timed versions of ACP, CSP and CCS in these papers differ concerning the degree to which time stamping is explicit in the notational format of the proposed process algebras. This in turn influences the form of equations, axiomatisations and the appearance of examples.

Another option is to divide time in slices indexed by natural numbers, to have an implicit or explicit time stamping mechanism that provides each action with the time slice in which it occurs and to have a

time order within each slice only. This has been worked out in ATP [NIS90], a process algebra that adds time slicing to a version of ACP based on action prefixing rather than sequential composition (this version was called aprACP in [BAB91b]). Further, [GRO90a] has extended ACP with time slices and [MOT89] have added these features to CCS. We propose to use the phrase *real time process algebra* for algebras that involve explicit or implicit time stamping with real numbers (or any dense subfield of the reals) and *discrete time process algebra* if an enumeration of time slices is used.

The objective of this paper is to extend ACP to a discrete time process algebra. This is done by adding a new process constructor $\sigma_d: P \to P$. The notation σ has been taken from [HER90], the subscript d draws attention to the discrete time setting. $\sigma_d(x)$ delays the process x till the next time slice. In addition to σ_d four auxiliary functions and an auxiliary sort are used. The booleans serve as auxiliary sort and the conditional operator $\_\lhd\_\rhd\_: P \times B \times P \to P$ is used. Further, an operator $1\gg_d\_: P \to P$ (unit time shift) reduces a process x to those options left if it idles one step (and to δ if it cannot idle). Then, the operator $\_\gg_d 1: P \to P$ restricts a process x to those options that do not involve initial idling. Finally, $D: P \to B$ determines whether or not a process can idle at all.

For practical application, the primitives of ACP_{dt} are too low level. So we have added the key operators of ATP [NSVR90, NIS90] to ACP_{dt}, thus obtaining ACP_{dt} + ATP (see [LYV92] for an extension of I/O automata with ATP operators). These features include: unit delay $\lfloor x \rfloor(y)$ and maximal progress composition $\oplus$. It turns out that these operators can be eliminated in the presence of those of ACP_{dt}. (It should be noted that ATP's $\oplus$ is already present as + in TCCS.) As ACP_{dt}, ATP and TCCS each use strong bisimulation equivalence to obtain a semantic domain, they may be considered as different (and intertranslatable) sets of generators for the same semantic world of processes. Some less important modifications with respect to ATP in the form of [NIS90] have been made. ACP_{dt} + ATP contains time stops (as does [MOT89]), a feature that is rejected on philosophical grounds in [NIS90]. In addition, the cancel mechanism has been omitted because that is covered automatically due to the presence of proper termination in ACP_{dt}.

Both the selection of ACP_{dt} + ATP and the casting of ATP in an ACP framework require some motivation. We have the following remarks on this.

1. ATP contains a combination of features (time outs, time posteriority, mode transfers) which turn it into a nontrivial extension of the untimed (symbolic) case. The features of ATP turn out to be of significant expressive power in comparison to [GRO90a], [GM91]. Moreover we found these features quite pleasant when dealing with examples.
2. The claim of [NIS90] that progress of time by itself should not be allowed to introduce non-determinism is convincing in our view. If time is represented by an action σ then $\sigma\cdot(x + y) = \sigma\cdot x + \sigma\cdot y$ is an appropriate axiom called time factorisation in [NIS90] ([GRO90a], [GM91] do not satisfy time factorisation).
3. ATP does not involve an action that represents the flow of time. This is a conceptual advantage as intuitively, flow of time is not an action, and the progress of an independent clock is not an observable system action.
4. Building ATP on top of ACP has several advantages. First of all, both termination and general sequential composition (which are in ACP but not in ATP) combine very well with ATP's key features. We hope that this is sufficiently illustrated by the protocol verifications in section 5. Secondly, the meta theory of ACP + ATP can be organised just as the meta theory of ACP (see e.g. [BAW90]) which allows to simply copy most of the results and proof techniques.

We design our discrete time process algebra in an incremental way, through stages BPA_{dt}, $BPA_{\delta dt}$, $PA_{\delta dt}$, ACP_{dt}, extensions of ACP_{dt}. This resembles the segmentation of ACP that can be found in [BAW90]. At each step we will motivate our choice and compare it with the sources of discrete time

process algebra that we have used: TCCS [MOT89], ACP_t [GRO90a], TPL [HER90], ATP [NIS90], CCSS [GM91]. Some of the axioms we use already appear in [BAB91b].

2. REAL TIME PROCESS ALGEBRA WITH INITIAL ABSTRACTION OPERATOR.

This paper intends to present discrete time process algebra in a clear connection with real time process algebra. To this end, we will first outline the axiom system $ACP\rho\sqrt{I}$. It describes the real time process algebra of [BAB91a] while integrating absolute and relative time notation using the initial abstraction operator $\sqrt{}$ that was introduced in [BAB92]. For $ACP\rho\sqrt{I}$, a bisimulation model M_A^* is outlined (here, A denotes the underlying set of atomic actions). This model is introduced in a sketchy way. Its relevance lies in the clear operational meaning it provides for processes. It serves as a conceptual standard model.

Having available M_A^*, the various operators and constants of discrete time process algebra are defined in it. Then, as a subject, discrete time process algebra reduces to an investigation of certain reducts and subalgebras of M_A^* as well as their axiomatic description.

We start with a quick overview of the real time process algebra introduced in [BAB91a]. We follow the presentation of [BAB92]. We also use the initial abstraction operator introduced in [BAB92], which allows to present the absolute time and relative time versions of the theory of [BAB91a] in a unified framework. We give an operational semantics in the style of [BAB91a].

2.1 ATOMIC ACTIONS.

We start from a set A of (symbolic) atomic actions. The set A is a parameter of the theory. Further, we have a special constant δ, denoting inaction, the absence of any execution. In particular, δ cannot terminate. We put $A_\delta = A \cup \{\delta\}$.

The set of atomic actions with time parameter, AT is

$$AT = \{a(t) \mid a \in A_\delta, t \in \mathbb{R}_{\geq 0}\}.$$

In [BAB91a], we considered the use of locations, in order to express that actions can occur at the same time but at different locations. This led to the introduction of *multi-actions*. In [BAB92], we extended this concept by looking at actions parametrised by a point in real space. As we will not need multi-actions in discrete time process algebra (since within a time slice, $a \| b = a \cdot b + b \cdot a + a \mid b$, i.e. interleaving works), we will not consider multi-actions in this paper. The theory can be set up for multi-actions, just like we did in [BAB92], without any problems.

2.2 BASIC PROCESS ALGEBRA.

Process algebra (see [BEK84, BAW90]) starts from a given *action alphabet*, here AT. Elements of AT are constants of the sort P of *processes*. The process $a(t)$ can let time progress until t, will then execute action a at time t, and then terminate successfully. The process $\delta(t)$ can also let time progress until t, but then nothing more is possible (in particular, time cannot progress anymore). Therefore, the δ actions are often called *time stops*.

Real Time Basic Process Algebra with Deadlock ($BPA\rho\delta$) has two binary operators $+, \cdot: P \times P \to P$; $+$ stands for alternative composition and $\cdot$ for sequential composition. Moreover, there is the additional operator $\gg: \mathbb{R}_{\geq 0} \times P \to P$, the *initialisation operator*. This operator was called the time shift

operator in [BAB91a]. $t \gg X$ denotes the process X starting at time t. This means that all actions that have to be performed at or before time t are turned into deadlocks because their execution has been delayed too long.

BPA$\rho\delta$ has the axioms from table 1 and 2 ($a \in A_\delta$).

$X + Y = Y + X$	A1
$(X+ Y) + Z = X + (Y + Z)$	A2
$X + X = X$	A3
$(X + Y)\cdot Z = X\cdot Z + Y\cdot Z$	A4
$(X\cdot Y)\cdot Z = X\cdot(Y\cdot Z)$	A5
$X + \delta = X$	A6
$\delta\cdot X = \delta$	A7

TABLE 1. BPA$_\delta$.

$a(0) = \delta(0) = \delta$	ATA1
$\delta(t) \cdot X = \delta(t)$	ATA2
$t < r \Rightarrow \delta(t) + \delta(r) = \delta(r)$	ATA3
$a(t) + \delta(t) = a(t)$	ATA4
$a(t) \cdot X = a(t) \cdot (t \gg X)$	ATA5
$t < r \Rightarrow t \gg a(r) = a(r)$	ATB1
$t \geq r \Rightarrow t \gg a(r) = \delta(t)$	ATB2
$t \gg (X + Y) = (t \gg X) + (t \gg Y)$	ATB3
$t \gg (X \cdot Y) = (t \gg X) \cdot Y$	ATB4

TABLE 2. Additional axioms of BPA$\rho\delta$.

The letter A in the names of the axioms in table 2 refers to *absolute time* (versions with relative time will be considered in the following). Next we consider parallel composition. Notice that we do not have a counterpart of PA ([BEK84]) in real time process algebra, because merge without communication is not an option here. Indeed, even if a and b do not communicate, the term $a(2) \| b(2)$ cannot be evaluated as $a(2)\cdot b(2) + b(2)\cdot a(2)$. Thus, we consider merge with communication straight away.

2.3 ALGEBRA OF COMMUNICATING PROCESSES.

In order to formulate communication between processes, we assume we have given a *communication function* $|$ on A_δ. This function $| : A_\delta \times A_\delta \to A_\delta$ is also a parameter of the theory, and should be commutative, associative and have δ as a neutral element (see axioms C1,2,3)

An axiomatization of parallel composition with communication uses the left merge operator $\lfloor$, the communication merge operator $|$, and the encapsulation operator ∂_H of [BEK84]. Moreover, two extra

auxiliary operators introduced in [BAB91a] are needed: the ultimate delay operator and the bounded initialization operator.

The *ultimate delay* operator U takes a process expression X, and returns an element of $\mathbb{R}_{\geq 0}$. The intended meaning is that X can idle before $U(X)$, but X can never reach time $U(X)$ or a later time by just idling. The *bounded initialization* operator (or *time out* operator) is also denoted by $\gg$, and is the counterpart of the operator with the same name that we saw in the axiomatization of $BPA\rho\delta$. With $X \gg t$ we denote the process X with its behaviour restricted to the extent that its first action must be performed at a time before $t \in \mathbb{R}_{\geq 0}$.

The axioms of ACPρ are in tables 1 through 3. In table 3, $H \subseteq A$, $a,b,c \in A_\delta$.

$U(a(t)) = t$		ATU1
$U(X + Y) = \max\{U(X), U(Y)\}$		ATU3
$U(X \cdot Y) = U(X)$		ATU4
$r \geq t \Rightarrow a(r) \gg t = \delta(t)$		ATB5
$r < t \Rightarrow a(r) \gg t = a(r)$		ATB6
$(X + Y) \gg t = (X \gg t) + (Y \gg t)$		ATB7
$(X \cdot Y) \gg t = (X \gg t) \cdot Y$		ATB8
$a \mid b = b \mid a$		C1
$a \mid (b \mid c) = (a \mid b) \mid c$		C2
$\delta \mid a = \delta$		C3
$t \neq s \Rightarrow a(t) \mid b(s) = \delta(\min(t,s))$		ATC1
$a(t) \mid b(t) = (a \mid b)(t)$		ATC2
$X \parallel Y = X \mathbin{\mathbb{L}} Y + Y \mathbin{\mathbb{L}} X + X \mid Y$		CM1
$X = r \gg Z \Rightarrow a(t) \mathbin{\mathbb{L}} X = (a(t) \gg U(X)) \cdot X$		ATCM2'
$Y = r \gg Z \Rightarrow a(t) \cdot X \mathbin{\mathbb{L}} Y = (a(t) \gg U(Y)) \cdot (X \parallel Y)$		ATCM3'
$(X + Y) \mathbin{\mathbb{L}} Z = X \mathbin{\mathbb{L}} Z + Y \mathbin{\mathbb{L}} Z$		CM4
$(a(t) \cdot X) \mid b(r) = (a(t) \mid b(r)) \cdot X$		CM5'
$a(t) \mid (b(r) \cdot X) = (a(t) \mid b(r)) \cdot X$		CM6'
$(a(t) \cdot X) \mid (b(r) \cdot Y) = (a(t) \mid b(r)) \cdot (X \parallel Y)$		CM7'
$(X + Y) \mid Z = X \mid Z + Y \mid Z$		CM8
$X \mid (Y + Z) = X \mid Y + X \mid Z$		CM9
$\partial_H(a) = a$	if $a \notin H$	D1
$\partial_H(a) = \delta$	if $a \in H$	D2
$\partial_H(a(t)) = (\partial_H(a))(t)$		ATD
$\partial_H(X + Y) = \partial_H(X) + \partial_H(Y)$		D3
$\partial_H(X \cdot Y) = \partial_H(X) \cdot \partial_H(Y)$		D4

TABLE 3. Remaining axioms of ACPρ.

Note that in [BAB91a], axioms ATCM2, ATCM3 appear that coincide with the axioms ATCM2', ATCM3' above except for the absence of the conditions. Because in the setting of ACPρ there, we have absolute time notation only, each process X satisfies $X = 0 \gg X$. We have that in the context of ACPρ, the new axioms are slightly weaker than the old, unprimed ones, but are in fact equivalent on finite process expressions. We have to add the conditions in order to deal with the following extension to processes that can also involve relative time notation. Notice that by substitution of $0 \gg Z$ for X resp. Y both conditional equations are turned into ordinary equations.

2.4 RELATIVE TIME NOTATION.

The following is based on [BAB92].

In some cases, relative time notation is profitable. In [BAB91a], we have introduced a relative time notation as follows: $a[t]$ denotes action a t time units after the previous atomic action or, if such action doesn't exist, after system initialisation. In this paper, we will deal with relative time in a different way. The basis for this approach to relative time notation is the *initial abstraction operator*. Let for $t \in \mathbb{R}_{\geq 0}$, $F(t)$ be a process in P, then

$$\surd t.F(t)$$

denotes a process that, when started at time r, proceeds as $F(r)$. Semantically, $\surd t.F(t)$ is just a function f from $\mathbb{R}_{\geq 0}$ into P that satisfies $f(t) = t \gg f(t)$ for all $t \geq 0$. We make things more precise in the following.

2.5 DEFINITION.

Let P be the sort of processes. Put

$$P^* = \{f : f\colon \mathbb{R}_{\geq 0} \to P \text{ such that } f(t) = t \gg f(t)\}.$$

We introduce the initial abstraction operator $\surd\colon T \times P^* \to P^*$, where T is a set of variables ranging over $\mathbb{R}_{\geq 0}$ called *time variables*. In the expression $\surd t.F$, free occurrences of t in F become bound.

2.6 AXIOMS.

All axioms of real time process algebra with integration presented before remain valid on the extended domain. On top of the old axioms, we have the extra axioms in table 4, governing the use of initial abstraction. We intend to axiomatise in such a way that all finite process expressions can be written in a form $\surd t.X$, with X not containing initial abstraction. In all axioms in table 4, it is assumed that t is not free in G. Axiom IA0 is the definition of the square bracket notation, axiom IA1 gives α-conversion, axiom IA2 achieves β-conversion, i.e. function application. IA4 gives the embedding of absolute time notation processes in the extended domain. IA5 is an extensionality axiom. We call the theory with axioms in tables 1-4 ACPρ$\surd$.

$a[r] = \surd t.\ a(t + r)$	IA0
$\surd s.G = \surd t.G[t/s]$	IA1
$s \gg \surd t.F = s \gg F[s/t]$	IA2
$\surd t.\ \surd r.\ F = \surd t.\ F[t/r]$	IA3
$G = \surd t.G$	IA4
$\forall t \geq 0\ t \gg X = t \gg Y \Rightarrow X = Y$	IA5
$(\surd t.F) + G = \surd t.(F + t \gg G)$	IA6
$(\surd t.F) \cdot G = \surd t.(F \cdot G)$	IA7
$(\surd t.F) \between G = \surd t.(F \between t \gg G)$	IA8
$G \between (\surd t.F) = \surd t.(t \gg G \between F)$	IA9
$(\surd t.F) \mid G = \surd t.(F \mid t \gg G)$	IA10
$G \mid (\surd t.F) = \surd t.(t \gg G \mid F)$	IA11
$\partial_H(\surd t.F) = \surd t.\partial_H(F)$	IA12
$(\surd t.F) \gg r = \surd t.(F \gg r)$	IA13
$U(F) = U(0 \gg F)$	IA14

TABLE 4. Axioms for initial abstraction.

2.7 EXAMPLES.

1. $a(3)\cdot(\surd t.\surd r.a(t + r + 1)) = a(3)\cdot(3 \gg \surd t.\surd r.a(t + r + 1)) = a(3)\cdot(3 \gg \surd r.a(r + 4)) =$
$= a(3)\cdot(3 \gg a(7)) = a(3)\cdot a(7).$
2. $a(t)\cdot b[r] = a(t)\cdot(\surd s.b(r + s)) = a(t)\cdot(t \gg \surd s.b(r + s)) = a(t)\cdot(t \gg b(r + t)) = a(t)\cdot b(r + t).$
3. We can calculate that $a(t)\cdot\partial_H(b[r]\cdot X \between Y) = a(t)\cdot\partial_H(b(t+r)\cdot X \between Y)$. Choose a fresh variable v (i.e. a time variable not occurring in X or Y). Then
$a(t)\cdot\partial_H(b[r]\cdot X \between Y) = a(t)\cdot\partial_H((\surd v.b(v+r))\cdot X \between Y) = a(t)\cdot\partial_H((\surd v.b(v+r)\cdot X) \between \surd v.Y) =$
$= a(t)\cdot\partial_H(\surd v.(b(v+r)\cdot X \between Y)) = a(t)\cdot\surd v.\partial_H(b(v+r)\cdot X \between Y) = a(t)\cdot(t \gg \surd v.\partial_H(b(v+r)\cdot X \between Y)) =$
$= a(t)\cdot(t \gg \partial_H(b(t+r)\cdot X \between Y)) = a(t)\cdot\partial_H(b(t+r)\cdot X \between Y).$
4. We can prove for all closed terms that $s \gg s \gg X = s \gg X$. We can use this to prove the following identity: $\surd t.\ F = \surd t.\ (t \gg F)$.
For, take $s \geq 0$, then $s \gg \surd t.F = s \gg F[s/t] = s \gg (s \gg F[s/t]) = s \gg (\surd t.\ t \gg F)$. The result follows by extensionality (IA5).

2.8 RIGIDITY.

We call a process X *rigid* iff $X = 0 \gg X$. The rigid processes in P^* correspond to the objects of P, now embedded into P^* by

$$p \mapsto \lambda t.\ t \gg p.$$

On rigid processes, the axioms ATCM2', ATCM3' reduce to their unprimed versions in ACPρ.

2.9 BOOLEANS.

A useful feature in the language is the addition of *conditionals* or *guards*. We follow the presentation of [BAB91c]. We use the auxiliary sort BOOL with constants true, false and add the ternary operator

$$.\lhd.\rhd.: P^* \times BOOL \times P^* \to BOOL.$$

Here, $x \lhd b \rhd y$ should be read as: **if** b **then** x **else** y. On occasion, we also this operator without the third component, the *guarded command* defined by

$$b :\to x = x \lhd b \rhd \delta.$$

Axioms for these operators are shown in table 5. In the case of expressions involving the initial abstraction operator, we will also encounter Boolean expressions parametrised by time variables, so we turn the sort BOOL into a function domain:

$$BOOL^* = \{f : f: \mathbb{R}_{\geq 0} \to BOOL\},$$

and we have the abstraction operator $\surd: T \times BOOL^* \to BOOL^*$. Boolean functions as $\wedge,\vee,\neg$ are defined on $BOOL^*$ pointwise. In table 5, $b,c \in BOOL^*$ with t not free in c.

The signature:

- sort BOOL*
- constants true, false
- functions $\neg: BOOL^* \to BOOL^*$
  $\wedge,\vee: BOOL^* \times BOOL^* \to BOOL^*$
  $<,>,\leq,\geq,=: \mathbb{R}_{\geq 0} \times \mathbb{R}_{\geq 0} \to BOOL^*$
  $\gg: \mathbb{R}_{\geq 0} \times BOOL^* \to BOOL^*$

$\neg$ true = false	$\neg$ false = true
$b \wedge$ false = false	$b \vee$ false = b
$b \wedge$ true = b	$b \vee$ true = true
$b \wedge c = c \wedge b$	$b \vee c = c \vee b$
$t \gg$ true = true	$t \gg$ false = false
$t \gg \surd r.\, b = t \gg b[t/r]$	
$\neg(\surd t.\, b) = \surd t.\, \neg b$	$(\surd t.\, b) \wedge c = \surd t.\, (b \wedge (t \gg c))$
$(\surd t.\, b) \vee c = \surd t.\, (b \vee (t \gg c))$	
$x \lhd$ true $\rhd y = x$	$x \lhd$ false $\rhd y = y$
$b :\to x = x \lhd b \rhd \delta$	$x \lhd b \rhd y = (b :\to x) + (\neg b :\to y)$
$(b \vee c) :\to x = (b :\to x) + (c :\to x)$	
$b :\to (c :\to x) = (b \wedge c) :\to x$	
$x \lhd \surd t.\, b \rhd y = \surd t.\, (t \gg x \lhd b \rhd t \gg y)$	
$t \gg (x \lhd b \rhd y) = t \gg x \lhd t \gg b \rhd t \gg y$	

TABLE 5. Axioms for conditionals.

2.10 EXAMPLE.

With the help of Boolean expressions, we can find normal forms for all closed process expressions of the form $\sqrt{t}.P$, with P using atomic actions, +, · and conditionals only. Thus, the operators $\|, \mathbb{L}, |$ can be eliminated. We just give two examples to illustrate this:

1. $a(3) = \sqrt{t}.\ (t < 3) :\rightarrow a(3)$.
2. $a(3) \| b[2] = \sqrt{t}.\ (a(3) \| b(t+2)) = \sqrt{t}.\ t \gg (a(3) \| b(t+2)) =$
$= \sqrt{t}.\ ((t < 1) :\rightarrow (b(t+2) \cdot a(3) + (t = 1) :\rightarrow (a \mid b)(3) + (t > 1) :\rightarrow a(3) \cdot b(t+2))$

2.11 INTEGRATION.

An extension of ACPρ√ (called ACPρ√I) that is very useful in applications is the extension with the integral operator, denoting a choice over a continuum of alternatives. That is, if V is a subset of $\mathbb{R}_{\geq 0}$, and v is a variable over $\mathbb{R}_{\geq 0}$, then $\int_{v \in V} P$ denotes the alternative composition of alternatives P[t/v] for $t \in V$ (expression P with nonnegative real t substituted for variable v). In this expression, a free occurrence of v in P becomes bound. Alpha conversion must be allowed to avoid name clashes. For more information, we refer the reader to [BAB91a] and [KLU91]. The untimed process algebra constants can be embedded in the timed setting as follows:

$$\underline{a} = \int_{t \geq 0} a(t), \qquad \underline{\delta} = \int_{t \geq 0} \delta(t).$$

Of course, we cannot give a complete axiomatisation of general integration because the general theory is undecidable, but the following axioms INT1-7 are very useful in derivations (for a complete axiomatisation of a subtheory, see [KLU91]). We also give an extra axiom for the initial abstraction operator.

$\int_{v \in \{t\}} P = P[t/v]$	INT1
$v \notin FV(P) \,\&\, V \neq \emptyset \Rightarrow \int_{v \in V} P = P$	INT2
$\int_{v \in \emptyset} P = \delta$	INT3
$\int_{v \in V \cup W} P = \int_{v \in V} P + \int_{v \in W} P$	INT4
$\int_{v \in V} (P + Q) = \int_{v \in V} P + \int_{v \in V} Q$	INT5
$v \notin FV(Q) \Rightarrow \int_{v \in V} (P \cdot Q) = (\int_{v \in V} P) \cdot Q$	INT6
for all $t \in V\ (P[t/v] = Q[t/v]) \Rightarrow \int_{v \in V} P = \int_{v \in V} Q$	INT7
$\int_{v \in V} \sqrt{t}.F = \sqrt{t}.(\int_{v \in V} F)$ if $t \neq v$	IA17

TABLE 6. Axioms for integration.

2.12 ELIMINATION.

In closed process expressions with a restricted form of integration, we can eliminate the operators $\|$, $\lfloor\!\lfloor$, $|$, ∂_H, $\gg$, $\gg$. [FOK92] explains in detail how such an elimination theorem can be obtained formally. They use the special case of prefixed integration. Generalisation of their results to a setting with initial abstraction seems unproblematic. This leads to normal forms of the form $\sqrt{t}.P$ with P using $a(t)$, $\delta(t)$, $+$, $\cdot$, conditionals and prefixed integration.

2.13 SEMANTICS.

Suppose we have an unspecified set of *states* S of cardinality $\leq 2^{\aleph_0}$ (when we use structured operational semantics further on, S will consist of closed process expressions plus the special symbol $\sqrt{}$). S contains a special state $\sqrt{}$ (the terminated state). A *real time transition system* over a set of actions A (not including δ) has domain $S \times \mathbb{R}_{\geq 0}$, has two relations on this domain:

$\text{idle} \subseteq S \times \mathbb{R}_{\geq 0} \times S \times \mathbb{R}_{\geq 0}$, notation $\langle s, t\rangle \rightarrow \langle s', t'\rangle$

$\text{step} \subseteq S \times \mathbb{R}_{\geq 0} \times A \times \mathbb{R}_{\geq 0} \times S \times \mathbb{R}_{\geq 0}$, notation $\langle s, t\rangle \xrightarrow{a(r)} \langle s', t'\rangle$,

and satisfies the following requirements:

0. there is a *root* $\langle s, 0\rangle$ for some $s \in S$; no transition starts from $\langle s', 0\rangle$ when $s' \neq s$
1. if $\langle s, t\rangle \rightarrow \langle s', t'\rangle$, then $t < t'$
2. if $\langle s, t\rangle \xrightarrow{a(r)} \langle s', t'\rangle$, then $t < r = t'$
3. if $\langle s, t\rangle \rightarrow \langle s', t'\rangle$ and $t < t^* < t'$, then there is $s^* \in S$ with $\langle s, t\rangle \rightarrow \langle s^*, t^*\rangle \rightarrow \langle s', t'\rangle$
4. if $\langle s, t\rangle \xrightarrow{a(r)} \langle s', t'\rangle$, and $t < t^* < r$, then there is $s^* \in S$ with $\langle s, t\rangle \rightarrow \langle s^*, t^*\rangle \xrightarrow{a(r)} \langle s', t'\rangle$
5. if $\langle s, t\rangle \rightarrow \langle s^*, t^*\rangle \rightarrow \langle s', t'\rangle$, then $\langle s, t\rangle \rightarrow \langle s', t'\rangle$
6. if $\langle s, t\rangle \rightarrow \langle s^*, t^*\rangle \xrightarrow{a(r)} \langle s', t'\rangle$, then $\langle s, t\rangle \xrightarrow{a(r)} \langle s', t'\rangle$
7. no transition starts from a state $\langle \sqrt{}, t\rangle$.

Let us call the set of such transition systems RTTS.

In order to deal with initial abstraction, we have to extend this set to the function space

$$\text{RTTS}^* = \{f : f: \mathbb{R}_{\geq 0} \rightarrow \text{RTTS}\}.$$

We can call elements of RTTS^* *type 1 transition systems.*

2.14 STRUCTURED OPERATIONAL SEMANTICS.

Now we interpret the constants and operators of $\text{ACP}\rho\sqrt{I}$ in the semantic domain of type 1 transition systems. We define the mapping $I^*: P^* \rightarrow \text{RTTS}^*$ by means of a mapping $I: P \rightarrow \text{RTTS}$. We have the identity $I^*(F) = \lambda t.\ I(t \gg F)$ (t not free in F),

so it remains to define I. The simplest way to define I is by means of structured operational semantics. As set of states we can take the set of closed process expressions plus $\sqrt{}$, the root of $I(x)$ is $\langle x, 0\rangle$ and the transitions are exactly those that can be derived on the basis of the rules in the following table 7. On the basis of these rules, the operators can also be directly defined on real time transition systems in a straightforward way. We have $t,s,r \in \mathbb{R}_{\geq 0}$, $x,x',y,y' \in S - \{\sqrt{}\}$, $a,b,c \in A$.

$t < r \Rightarrow \langle a(r), t\rangle \xrightarrow{a(r)} \langle \surd, r\rangle$

$t < s < r \Rightarrow \langle a(r), t\rangle \rightarrow \langle a(r), s\rangle$

$t < s < r \Rightarrow \langle \delta(r), t\rangle \rightarrow \langle \delta(r), s\rangle$

$$\frac{\langle x,t\rangle \xrightarrow{a(r)} \langle x',r\rangle}{\langle x+y,t\rangle \xrightarrow{a(r)} \langle x',r\rangle,\ \langle y+x,t\rangle \xrightarrow{a(r)} \langle x',r\rangle}$$

$$\frac{\langle x,t\rangle \xrightarrow{a(r)} \langle \surd,r\rangle}{\langle x+y,t\rangle \xrightarrow{a(r)} \langle \surd,r\rangle,\ \langle y+x,t\rangle \xrightarrow{a(r)} \langle \surd,r\rangle}$$

$$\frac{\langle x,t\rangle \rightarrow \langle x,r\rangle}{\langle x+y,t\rangle \rightarrow \langle x+y,r\rangle,\ \langle y+x,t\rangle \rightarrow \langle y+x,r\rangle}$$

$$\frac{\langle x,t\rangle \xrightarrow{a(r)} \langle x',r\rangle}{\langle x\cdot y,t\rangle \xrightarrow{a(r)} \langle x'\cdot y,r\rangle} \qquad \frac{\langle x,t\rangle \rightarrow \langle x,r\rangle}{\langle x\cdot y,t\rangle \rightarrow \langle x\cdot y,r\rangle}$$

$$\frac{\langle x,t\rangle \xrightarrow{a(r)} \langle \surd,r\rangle}{\langle x\cdot y,t\rangle \xrightarrow{a(r)} \langle y,r\rangle}$$

$t < r < s \Rightarrow \langle s \gg x, t\rangle \rightarrow \langle s \gg x, r\rangle$

$$\frac{\langle x,t\rangle \xrightarrow{a(r)} \langle x',r\rangle,\ r>s}{\langle s\gg x,t\rangle \xrightarrow{a(r)} \langle x',r\rangle} \qquad \frac{\langle x,t\rangle \rightarrow \langle x,r\rangle}{\langle s\gg x,t\rangle \rightarrow \langle s\gg x,r\rangle}$$

$$\frac{\langle x,t\rangle \xrightarrow{a(r)} \langle \surd,r\rangle,\ r>s}{\langle s\gg x,t\rangle \xrightarrow{a(r)} \langle \surd,r\rangle}$$

$$\frac{\langle x,t\rangle \xrightarrow{a(r)} \langle x',r\rangle,\ \langle y,t\rangle \to \langle y,r\rangle}{\langle x \| y,t\rangle \xrightarrow{a(r)} \langle x' \| y,r\rangle,\ \langle y \| x,t\rangle \xrightarrow{a(r)} \langle y \| x',r\rangle,\ \langle x \mathbb{L} y,t\rangle \xrightarrow{a(r)} \langle x' \| y,r\rangle}$$

$$\frac{\langle x,t\rangle \xrightarrow{a(r)} \langle \surd,r\rangle,\ \langle y,t\rangle \to \langle y,r\rangle}{\langle x \| y,t\rangle \xrightarrow{a(r)} \langle y,r\rangle,\ \langle y \| x,t\rangle \xrightarrow{a(r)} \langle y,r\rangle,\ \langle x \mathbb{L} y,t\rangle \xrightarrow{a(r)} \langle y,r\rangle}$$

$$\frac{\langle x,t\rangle \to \langle x,r\rangle,\ \langle y,t\rangle \to \langle y,r\rangle}{\langle x \| y,t\rangle \to \langle x \| y,r\rangle,\ \langle x \mathbb{L} y,t\rangle \to \langle x \mathbb{L} y,r\rangle}$$

$$\frac{\langle x,t\rangle \xrightarrow{a(r)} \langle x',r\rangle,\ \langle y,t\rangle \xrightarrow{b(r)} \langle y',r\rangle,\ a \mid b=c\neq\delta}{\langle x \| y,t\rangle \xrightarrow{c(r)} \langle x' \| y',r\rangle,\ \langle x \mid y,t\rangle \xrightarrow{c(r)} \langle x' \| y',r\rangle}$$

$$\frac{\langle x,t\rangle \xrightarrow{a(r)} \langle x',r\rangle,\ \langle y,t\rangle \xrightarrow{b(r)} \langle \surd,r\rangle,\ a \mid b=c\neq\delta}{\begin{array}{l}\langle x \| y,t\rangle \xrightarrow{c(r)} \langle x',r\rangle,\ \langle y \| x,t\rangle \xrightarrow{c(r)} \langle x',r\rangle,\\ \langle x \mid y,t\rangle \xrightarrow{c(r)} \langle x',r\rangle,\ \langle y \mid x,t\rangle \xrightarrow{c(r)} \langle x',r\rangle\end{array}}$$

$$\frac{\langle x,t\rangle \xrightarrow{a(r)} \langle \surd,r\rangle,\ \langle y,t\rangle \xrightarrow{b(r)} \langle \surd,r\rangle,\ a \mid b=c\neq\delta}{\langle x \| y,t\rangle \xrightarrow{c(r)} \langle \surd,r\rangle,\ \langle x \mid y,t\rangle \xrightarrow{c(r)} \langle \surd,r\rangle}$$

$$\frac{\langle x,t\rangle \to \langle x,r\rangle,\ r<s}{\langle x \gg s,t\rangle \to \langle x \gg s,r\rangle} \qquad \frac{\langle x,t\rangle \xrightarrow{a(r)} \langle x',r\rangle,\ r<s}{\langle x \gg s,t\rangle \xrightarrow{a(r)} \langle x',r\rangle}$$

$$\frac{\langle x,t\rangle \xrightarrow{a(r)} \langle \surd,r\rangle,\ r<s}{\langle x \gg s,t\rangle \xrightarrow{a(r)} \langle \surd,r\rangle}$$

$$\frac{\langle x,t\rangle \xrightarrow{a(r)} \langle x',r\rangle,\ a\notin H}{\langle \partial_H(x),t\rangle \xrightarrow{a(r)} \langle \partial_H(x'),r\rangle} \qquad \frac{\langle x,t\rangle \to \langle x,r\rangle}{\langle \partial_H(x),t\rangle \to \langle \partial_H(x),r\rangle}$$

$$\frac{\langle x,t\rangle \xrightarrow{a(r)} \langle \surd,r\rangle,\ a\notin H}{\langle \partial_H(x),t\rangle \xrightarrow{a(r)} \langle \surd,r\rangle}$$

$$\frac{\langle x,t\rangle \xrightarrow{a(r)} \langle x',r\rangle}{\langle x \lhd \text{true} \rhd y,t\rangle \xrightarrow{a(r)} \langle x',r\rangle,\ \langle y \lhd \text{false} \rhd x,t\rangle \xrightarrow{a(r)} \langle x',r\rangle}$$

$$\frac{\langle x,t\rangle \xrightarrow{a(r)} \langle \surd,r\rangle}{\langle x \lhd \text{true} \rhd y,t\rangle \xrightarrow{a(r)} \langle \surd,r\rangle,\ \langle y \lhd \text{false} \rhd x,t\rangle \xrightarrow{a(r)} \langle \surd,r\rangle}$$

$$\frac{\langle x,t\rangle \to \langle x,r\rangle}{\langle x \lhd \text{true} \rhd y,t\rangle \to \langle x \lhd \text{true} \rhd y,r\rangle,\ \langle y \lhd \text{false} \rhd x,t\rangle \to \langle y \lhd \text{false} \rhd x,r\rangle}$$

$$\frac{\langle x(u),t\rangle \to \langle x(u),r\rangle,\ u \in V}{\langle \int_{v\in V} x(v),t\rangle \to \langle \int_{v\in V} x(v),r\rangle}$$

$$\frac{\langle x(u),t\rangle \xrightarrow{a(r)} \langle x',r\rangle \quad u \in V}{\langle \int_{v\in V} x(v),t\rangle \xrightarrow{a(r)} \langle x',r\rangle}$$

$$\frac{\langle x(u),t\rangle \xrightarrow{a(r)} \langle \surd,r\rangle \quad u \in V}{\langle \int_{v\in V} x(v),t\rangle \xrightarrow{a(r)} \langle \surd,r\rangle}$$

TABLE 7. Structured operational semantics.

All that remains is to define the ultimate delay operator U on RTTS*. If $f: \mathbb{R}_{\geq 0} \to$ RTTS, then $U(f) = U(f(0))$, and if $R \in$ RTTS, then

$$U(R) = \sup\{t \in \mathbb{R}_{\geq 0} : \text{there is } s \in S \text{ with } \text{root}(R) \to \langle s, t\rangle\}.$$

2.15 BISIMULATIONS.

A *bisimulation* on RTTS is a binary relation R on $S \times \mathbb{R}_{\geq 0}$ such that ($a \in A$):

i. whenever $R(\langle s, t\rangle, \langle s', t'\rangle)$ then $t = t'$

ii. for each p and q with $R(p, q)$: if there is a step $a(t)$ possible from p to p', then there is $q' \in S \times \mathbb{R}_{\geq 0}$ such that $R(p', q')$ and there is a step $a(t)$ possible from q to q'.

iii. for each p and q with $R(p, q)$: if there is a step $a(t)$ possible from q to q', then there is $p' \in S \times \mathbb{R}_{\geq 0}$ such that $R(p', q')$ and there is a step $a(t)$ possible from p to p'.

iv. for each p and q with $R(p, q)$: a termination step $a(t)$ to $\langle \surd, t\rangle$ is possible from p iff it is possible from q.

v. for each p and q with $R(p, q)$: if there is an idle step possible from p to p', then there is $q' \in S \times \mathbb{R}_{\geq 0}$ such that $R(p', q')$ and there is an idle step possible from q to q'.

vi. for each p and q with $R(p, q)$: if there is an idle step possible from q to q', then there is $p' \in S \times \mathbb{R}_{\geq 0}$ such that $R(p', q')$ and there is an idle step possible from p to p'.

We say expressions p and q are *bisimilar*, denoted $p \leftrightarrow q$, if there exists a bisimulation with $R(p,q)$. We can show that bisimulation is a congruence relation on real time transition systems, and that real time transition systems modulo bisimulation determine a model for ACPρI. This model also contains solutions of recursive process definitions, but that is an issue that is outside the scope of the present paper. We obtain a model of ACPρI that we will call M_A.

This definition of bisimulation naturally extends to a definition on type 1 real time transition systems: two elements $f,g \in RTTS^*$ are bisimilar if for all $t \geq 0$, $f(t)$ and $g(t)$ are bisimilar. This gives us a model M_A^* of type 1 real time transition systems modulo bisimulation.

2.16 STANDARD INITIALISATION AXIOMS.

We present some axioms that are useful in the calculations to come, that hold in the model we described, and that can be derived from the theory for all closed process expressions. Thus, these axioms have the same status as the so-called *Standard Concurrency* axioms of [BEK84].

$t \leq r \Rightarrow t \gg (x \gg r) = (t \gg x) \gg r$
$t \leq r \Rightarrow t \gg (r \gg x) = r \gg x$
$t \geq r \Rightarrow t \gg (x \gg r) = \delta(t)$
$t \geq r \Rightarrow (t \gg x) \gg r = \delta(r)$

TABLE 8. Standard initialisation axioms.

3. THE SYNTAX OF DISCRETE TIME PROCESS ALGEBRA.

3.1 SIGNATURE.

The signature of discrete time process algebra provides constants and operators parametrised by natural numbers rather than real numbers. We add the following ingredients to real time process algebra.

- constants

$a(n)$	a in the n-th time slice ($a \in A$, $n \in \mathbb{N}$)
$\underline{a}$	a in the current time slice (= $a[1]$)
$a[n]$	a in the $(n-1)$-st time slice after the current one ($n \in \mathbb{N}$)
$\delta(n)$	time stop at the end of the n-th time slice ($n \in \mathbb{N}$)
$\underline{\delta}$	δ at the end of the current time slice (= $\delta[1]$)
$\delta[n]$	δ at the end of the $(n-1)$-st time slice after the current one ($n \in \mathbb{N}$)
$\underline{a}$	delayable a ([BAB91a])
$\underline{\delta}$	livelock ([BAB91a])
T	discrete time step ([GRO90a])

- functions

$\gg_d 1: P^* \rightarrow P^*$	time out in the current time slice
$1 \gg_d: P^* \rightarrow P^*$	initialisation in the following time slice
$\sigma_d: P^* \rightarrow P^*$	discrete time unit delay

$\sigma^t: P^* \to P^*$	real time delay by time $t \in \mathbb{R}_{\geq 0}$ ([MOT89])
$D: P^* \to BOOL^*$	delayability
$\surd_d : N \times P^* \to P^*$	discrete time initialisation ([HER90])
$\oplus: P^* \times P^* \to P^*$	strong choice
$\lfloor . \rfloor(.): P^* \times P^* \to P^*$	unit delay
$\lfloor . \rfloor^n(.): P^* \times P^* \to P^*$	start delay within n ($n \in \mathbb{N}$)
$\lfloor . \rfloor^\omega: P^* \to P^*$	unbounded start delay
$\lceil . \rceil^n(.): P^* \times P^* \to P^*$	execution delay.

The last five operators are taken from ATP [NIS90]. We will call these the Nicollin-Sifakis functions for discrete time process algebra.

3.2 DEFINITIONS.

Now we give the definitions of the new signature ingredients in table 9, except for the execution delay $\lceil . \rceil^n(.)$. For $t \in \mathbb{R}$, $\lfloor t \rfloor$ is the floor of t, the largest integer less than or equal to t. We will also on occasion need the largest integer less than t, and define $\lfloor t \rfloor' = \max\{0, \max\{n \in \mathbb{Z} : n < t\}\}$.

3.3 COMMENTS.

We will make several comments concerning the definitions in table 9.

1. The definition of τ uses a special atomic action t, that satisfies $t \mid t = t$ and $t \mid a = \delta$ for $a \neq t$. τ is Groote's time step of [GRO90a]. Thus, Groote's t becomes a 'grote t', avoiding confusion for Dutchmen.
2. We follow [MOT89] by taking $\underline{\delta}$ (0 in [MOT89]) to be a time stop. $\underline{\delta}$ is like δ in [GRO90a] and like nil in [GM91]. We notice that TPL has no constant that introduces a time stop, but $\underline{\delta}$ seems to be equivalent to an infinite τ-loop in TPL [HER90]. Although time stops do not occur in actual computing, we see no objection to their presence in the calculus. So we so not follow ATP's design rationale, which excludes time stops on a priori grounds. $\underline{a}$ is like atomic actions in ATP.
3. We have a (unit) delay operator σ_d rather than an action representing the progress of time. The name σ for the delay operator has been taken from TPL [HER90] where σ is used to name the action that represents delay. TCCS [MOT89] has a unary operator for delay just like we have. ATP [NIS90] also introduces delay via an operator, be it a binary one ($\lfloor . \rfloor(.)$). $\underline{a}$ models the delayable atoms of TPL.
4. Our + has the interpretation of weak choice in TCCS [MOT89].
5. The notation for the delayability predicate has been taken from TCCS (where it denotes a slightly different operator, however).

$$\underline{a} = \int_{t>0} a(t)$$

$$\underline{\delta} = \int_{t>0} \delta(t)$$

$$a(0) = \delta$$

$$a(n+1) = n \gg (\underline{a} \gg (n+1))$$

$$a[0] = \delta$$

$$\underline{a} = a[1]$$

$$a[1] = \sqrt{t}.\ \underline{a} \gg \lfloor t+1 \rfloor$$

$$a[n+2] = \sigma_d(a[n+1])$$

$$\delta(n) = \delta(n)$$

$$\delta[0] = \delta$$

$$\underline{\delta} = \delta[1]$$

$$\delta[1] = \sqrt{t}.\ \delta(\lfloor t+1 \rfloor)$$

$$\delta[n+2] = \sigma_d(\delta[n+1])$$

$$T = \sqrt{r}.\ t(\lfloor r+1 \rfloor)$$

$$x \gg_d 1 = \sqrt{t}.\ (\lfloor t \rfloor \gg x) \gg \lfloor t+1 \rfloor$$

$$1 \gg_d x = \sqrt{t}.\ \lfloor t+1 \rfloor' \gg (\lfloor t \rfloor' \gg x)$$

$$\sigma_d(x) = \sqrt{t}.\ \lfloor t+1 \rfloor \gg x$$

$$\sigma^t(x) = \sqrt{t}.\ (t+r) \gg x$$

$$D(x) = \lambda t.\ U(\lfloor t \rfloor \gg x) \geq \lfloor t+1 \rfloor$$

$$\sqrt{_d}n.\ X = \sqrt{t}.\ X(\lfloor t \rfloor / n)$$

$$x \oplus y = \sqrt{t}.\ \Big(\lfloor t \rfloor \gg x + y \gg \lfloor t+1 \rfloor + \sum_{i=0}^{\infty} \lfloor t+i+1 \rfloor \gg (\lfloor t \rfloor \gg x+y) \gg \lfloor t+i+2 \rfloor \lhd D(\lfloor t \rfloor \gg x)(\lfloor t+i \rfloor) \wedge D(\lfloor t \rfloor \gg y)(\lfloor t+i \rfloor) \rhd \underline{\delta}\Big)$$

$$\lfloor x \rfloor(y) = x \gg_d 1 + \sigma_d(y)$$

$$\lfloor x \rfloor^0(y) = y$$

$$\lfloor x \rfloor^{n+1}(y) = \lfloor x \rfloor(\lfloor x \rfloor^n(y))$$

$$\lfloor x \rfloor^{\omega} = \sqrt{t}.\ \sum_{i=0}^{\infty} \lfloor t+i \rfloor \gg x \gg \lfloor t+i+1 \rfloor$$

TABLE 9. Discrete time.

3.4 EXECUTION DELAY.

In order to give the definition of the execution delay operator $\lceil . \rceil^n(.)$, we first need to define an extra real time operator $\lceil . \rceil^t_\rho(.)$. We want to stick to the ATP notation for the execution delay operator, and so we use a distinguishing subscript ρ for the real time variant. We have the axioms in table 10.

$\lceil a(r) \rceil^t_\rho(x) = a(r) \lhd r < t \rhd t \gg (0 \gg x)$
$\lceil a(r) \cdot x \rceil^t_\rho(y) = a(r) \cdot \lceil r \gg x \rceil^t_\rho(y) \lhd r < t \rhd t \gg (0 \gg y)$
$\lceil x + y \rceil^t_\rho(z) = \lceil x \rceil^t_\rho(z) + \lceil y \rceil^t_\rho(z)$
$\lceil \sqrt{s}.\ x \rceil^t_\rho(y) = \lceil 0 \gg x[0/s] \rceil^t_\rho(y)$
$\lceil x \rceil^0(y) = y$
$\lceil x \rceil^{n+1}(y) = \sqrt{r}. \lceil r \gg x \rceil^{\lfloor r+n+1 \rfloor}_\rho (\lfloor r+n+1 \rfloor \gg y)$

TABLE 10. Execution delay.

4. PROPERTIES AND THEIR PROOFS.

4.1 LEMMA: $n \gg a(m) = a(m) \lhd n < m \rhd \delta(n)$.

PROOF: By induction on m.

Case 1: induction basis, $m = 0$: $n \gg a(0) = n \gg \delta = \delta(n) = \delta(n) \lhd n = 0 \rhd \delta(n) =$

$= a(0) \lhd n \le 0 \rhd \delta(n)$.

Case 2: induction step: $n \gg a(m+1) = n \gg (m \gg (\underline{a} \gg m+1)) = n \gg (m \gg (\int_{t>0} a(t) \gg m+1) =$

$$= \int_{t>0} n \gg (m \gg (a(t) \gg m+1)) = \int_{t>0} m \gg (a(t) \gg m+1) \lhd n \le m \rhd \int_{t>0} \delta(n) =$$

$$= m \gg (\underline{a} \gg m+1) \lhd n \le m \rhd \delta(n) = a(m+1) \lhd n < m+1 \rhd \delta(n).$$

4.2 PROPOSITION: $\sigma_d(x + y) = \sigma_d(x) + \sigma_d(y)$.

COMMENT: This identity is called *time factorisation* in [NSVR90]. Also TCCS [MoT89] and TPL [HER90] have time factorisation.

PROOF: $\sigma_d(x + y) = \sqrt{t}. \lfloor t+1 \rfloor \gg (x + y) = \sqrt{t}. (\lfloor t+1 \rfloor \gg x + \lfloor t+1 \rfloor \gg y) =$

$= \sqrt{t}. \sqrt{r}. (\lfloor r+1 \rfloor \gg x + \lfloor t+1 \rfloor \gg y) = \sqrt{t}. ((\sqrt{r}. \lfloor r+1 \rfloor \gg x) + \lfloor t+1 \rfloor \gg y) =$

$= \sqrt{r}. \lfloor r+1 \rfloor \gg x + \sqrt{t}. \lfloor t+1 \rfloor \gg y = \sigma_d(x) + \sigma_d(y)$.

4.3 PROPOSITION: $\sigma_d(x \cdot y) = \sigma_d(x) \cdot y$.

PROOF: $\sigma_d(x \cdot y) = \sqrt{t}. \lfloor t+1 \rfloor \gg (x \cdot y) = \sqrt{t}. (\lfloor t+1 \rfloor \gg x) \cdot y = (\sqrt{t}. \lfloor t+1 \rfloor \gg x) \cdot y = \sigma_d(x) \cdot y$.

4.4. PROPOSITION: $\lfloor x \rfloor^\omega = \lfloor x \rfloor (\lfloor x \rfloor^\omega)$

PROOF: $\lfloor x \rfloor^\omega = \sqrt{t}. \sum_{i=0}^{\infty} \lfloor t+i \rfloor \gg x \gg \lfloor t+i+1 \rfloor = \sqrt{t}. \lfloor t \rfloor \gg x \gg \lfloor t+1 \rfloor + \sum_{i=1}^{\infty} \sqrt{t}. \lfloor t+i \rfloor \gg x \gg \lfloor t+i+1 \rfloor$

$$= x \gg_d 1 + \sum_{i=0}^{\infty} \sqrt{t}.\, \lfloor t+i+1 \rfloor \gg x \gg \lfloor t+i+2 \rfloor =$$

$$= x \gg_d 1 + \sum_{i=0}^{\infty} \sqrt{t}.\, \lfloor t+1 \rfloor \gg (\sqrt{r}.\, \lfloor r+i \rfloor \gg x \gg \lfloor r+i+1 \rfloor) =$$

$$= x \gg_d 1 + \sqrt{t}.\, \lfloor t+1 \rfloor \gg (\sqrt{r}.\, \sum_{i=0}^{\infty} \lfloor r+i \rfloor \gg x \gg \lfloor r+i+1 \rfloor) =$$

$$= x \gg_d 1 + \sigma_d(\lfloor x \rfloor^{\omega}) = \lfloor x \rfloor(\lfloor x \rfloor^{\omega})$$

4.5 PROPOSITION: $x \oplus y = (x+y) \gg_d 1 + \sigma_d(1 \gg_d x \oplus 1 \gg_d y) \lhd D(x) \wedge D(y) \rhd \underline{\delta}$.

PROOF: $x \oplus y = \sqrt{t}.\, (\lfloor t \rfloor \gg (x + y) \gg \lfloor t+1 \rfloor) +$

$$+ \sqrt{t}.\, \sum_{i=0}^{\infty} \lfloor t+i+1 \rfloor \gg (\lfloor t \rfloor \gg (x+y)) \gg \lfloor t+i+2 \rfloor \lhd D(\lfloor t \rfloor \gg x)(\lfloor t+i \rfloor) \wedge D(\lfloor t \rfloor \gg y)(\lfloor t+i \rfloor) \rhd \underline{\delta} =$$

$$= (x+y) \gg_d 1 +$$

$$+ \sqrt{t}.\, \lfloor t+1 \rfloor \gg ((\lfloor t \rfloor \gg (x + y)) \gg \lfloor t+2 \rfloor) \lhd D(\lfloor t \rfloor \gg x)(\lfloor t \rfloor) \wedge D(\lfloor t \rfloor \gg y)(\lfloor t \rfloor) \rhd \underline{\delta} +$$

$$+ \sqrt{t}.\, \sum_{i=1}^{\infty} \lfloor t+1+i \rfloor \gg (\lfloor t \rfloor \gg (x+y)) \gg \lfloor t+2+i \rfloor \lhd D(\lfloor t \rfloor \gg x)(\lfloor t+1+i \rfloor) \wedge D(\lfloor t \rfloor \gg y)(\lfloor t+1+i \rfloor) \rhd \underline{\delta} =$$

$$= (x+y) \gg_d 1 +$$

$$+ \sqrt{t}.\, \lfloor t+1 \rfloor \gg \lfloor t \rfloor \gg (x + y) \gg \lfloor t+2 \rfloor) \lhd D(\lfloor t \rfloor \gg x) \wedge D(\lfloor t \rfloor \gg y) \rhd \underline{\delta} +$$

$$+ \sqrt{t}.\, (\sum_{i=1}^{\infty} \lfloor t+1+i \rfloor \gg (\lfloor t \rfloor \gg (x+y)) \gg \lfloor t+2+i \rfloor$$

$$\lhd D(\lfloor t+1 \rfloor \gg \lfloor t \rfloor \gg x)(\lfloor t+1+i \rfloor) \wedge D(\lfloor t+1 \rfloor \gg \lfloor t \rfloor \gg y)(\lfloor t+1+i \rfloor) \rhd \underline{\delta})$$

$$\lhd U(\lfloor t \rfloor \gg x) \geq \lfloor t+1 \rfloor \wedge U(\lfloor t \rfloor \gg y) \geq \lfloor t+1 \rfloor \rhd \delta(\lfloor t+1 \rfloor) =$$

$$= (x+y) \gg_d 1 +$$

$$+ \sqrt{t}.\, \lfloor t+1 \rfloor \gg \big((\lfloor t+1 \rfloor \gg \lfloor t \rfloor \gg (x + y) \gg \lfloor t+2 \rfloor) +$$

$$+ \sum_{i=1}^{\infty} \lfloor t+1+i \rfloor \gg (\lfloor t+1 \rfloor \gg \lfloor t \rfloor \gg (x + y)) \gg \lfloor t+2+i \rfloor$$

$$\lhd D(\lfloor t+1 \rfloor \gg \lfloor t \rfloor \gg x)(\lfloor t+1+i \rfloor) \wedge D(\lfloor t+1 \rfloor \gg \lfloor t \rfloor \gg y)(\lfloor t+1+i \rfloor) \rhd \underline{\delta}\big)$$

$$\lhd U(\lfloor t \rfloor \gg x) \geq \lfloor t+1 \rfloor \wedge U(\lfloor t \rfloor \gg y) \geq \lfloor t+1 \rfloor \rhd \delta(\lfloor t+1 \rfloor) =$$

$$= (x+y) \gg_d 1 +$$

$$+ \sqrt{t}.\, \lfloor t+1 \rfloor \gg \big(\lfloor t+1 \rfloor \gg \lfloor t+1 \rfloor \gg \lfloor t \rfloor \gg (x + y) \gg \lfloor t+2 \rfloor +$$

$$+\sum_{i=1}^{\infty}\lfloor t+1+i\rfloor \gg (\lfloor t+1\rfloor \gg \lfloor t+1\rfloor \gg \lfloor t\rfloor \gg (x+y)) \gg \lfloor t+2+i\rfloor$$

$$\lhd D(\lfloor t+1\rfloor \gg \lfloor t+1\rfloor \gg \lfloor t\rfloor \gg x)(\lfloor t+1+i\rfloor) \wedge D(\lfloor t+1\rfloor \gg \lfloor t+1\rfloor \gg \lfloor t\rfloor \gg y)(\lfloor t+1+i\rfloor) \rhd \underline{\delta})$$

$$\lhd U(\lfloor t\rfloor \gg x) \geq \lfloor t+1\rfloor \wedge U(\lfloor t\rfloor \gg y) \geq \lfloor t+1\rfloor \rhd \delta(\lfloor t+1\rfloor) =$$

$$= (x+y) \gg_d 1 +$$

$$+ \surd t.\ \lfloor t+1\rfloor \gg (\lfloor t+1\rfloor \gg ((1 \gg_d x) + (1 \gg_d y)) \gg \lfloor t+2\rfloor) +$$

$$+\sum_{i=0}^{\infty}\lfloor t+2+i\rfloor \gg (\lfloor t+1\rfloor \gg ((1 \gg_d x) + (1 \gg_d y))) \gg \lfloor t+3+i\rfloor$$

$$\lhd D(\lfloor t+1\rfloor \gg (1 \gg_d x))(\lfloor t+1+i\rfloor) \wedge D(\lfloor t+1\rfloor \gg (1 \gg_d y))(\lfloor t+1+i\rfloor) \rhd \underline{\delta})$$

$$\lhd U(\lfloor t\rfloor \gg x) \geq \lfloor t+1\rfloor \wedge U(\lfloor t\rfloor \gg y) \geq \lfloor t+1\rfloor \rhd \delta(\lfloor t+1\rfloor) =$$

$$= (x+y) \gg_d 1 +$$

$$+ \surd t.\ \lfloor t+1\rfloor \gg (\surd r.\ (\lfloor r\rfloor \gg ((1 \gg_d x) + (1 \gg_d y)) \gg \lfloor r+1\rfloor) +$$

$$+\sum_{i=0}^{\infty}\lfloor r+i+1\rfloor \gg (\lfloor r\rfloor \gg ((1 \gg_d x) + (1 \gg_d y))) \gg \lfloor r+i+2\rfloor$$

$$\lhd D(\lfloor r\rfloor \gg (1 \gg_d x))(\lfloor r+i\rfloor) \wedge D(\lfloor r\rfloor \gg (1 \gg_d y))(\lfloor r+i\rfloor) \rhd \underline{\delta})$$

$$\lhd t \gg (D(x) \wedge D(y)) \rhd t \gg \underline{\delta} =$$

$$= (x+y) \gg_d 1 +$$

$$+ \surd t.\ (\lfloor t+1\rfloor \gg (1 \gg_d x \oplus 1 \gg_d y)$$

$$\lhd t \gg (D(x) \wedge D(y)) \rhd t \gg \underline{\delta}) =$$

$$= (x+y) \gg_d 1 + (\surd t.\ \lfloor t+1\rfloor \gg (1 \gg_d x \oplus 1 \gg_d y)) \lhd D(x) \wedge D(y) \rhd \underline{\delta} =$$

$$= (x+y) \gg_d 1 + \sigma_d(1 \gg_d x \oplus 1 \gg_d y) \lhd D(x) \wedge D(y) \rhd \underline{\delta}.$$

4.6 PROPOSITION: $\sigma_d(x) \gg_d 1 = \underline{\delta}$.

PROOF: $\sigma_d(x) \gg_d 1 = (\surd t.\ \lfloor t+1\rfloor \gg x) \gg_d 1 = \surd r.\ \lfloor r\rfloor \gg (\surd t.\ \lfloor t+1\rfloor \gg x) \gg \lfloor r+1\rfloor =$

$$= \surd r.\ \lfloor r\rfloor \gg (\lfloor\lfloor r\rfloor+1\rfloor \gg x) \gg \lfloor r+1\rfloor = \surd r.\ \lfloor r\rfloor \gg (\lfloor r+1\rfloor \gg x \gg \lfloor r+1\rfloor) =$$

$$= \surd r.\ \lfloor r\rfloor \gg \delta(\lfloor r+1\rfloor) = \surd r.\ \delta(\lfloor r+1\rfloor) = \surd r.\ \underline{\delta} \gg \lfloor r+1\rfloor = \underline{\delta}.$$

4.7 PROPOSITION: $\lceil \underline{a} \rceil^{n+1}(x) = \underline{a}$.

PROOF: $\lceil \underline{a} \rceil^{n+1}(x) = \surd r.\ \lceil r \gg \underline{a} \rceil_{\rho}^{\lfloor r+n+1 \rfloor}(\lfloor r+n+1\rfloor \gg x) = \surd r.\ \lceil \int_{t \in (r, \lfloor r+1 \rfloor)} a(t) \rceil_{\rho}^{\lfloor r+n+1 \rfloor}(\lfloor r+n+1\rfloor \gg x) =$

$$\surd r.\ \int_{t \in (r, \lfloor r+1 \rfloor)} \lceil a(t) \rceil_{\rho}^{\lfloor r+n+1 \rfloor}(\lfloor r+n+1\rfloor \gg x) = \surd r.\ \int_{t \in (r, \lfloor r+1 \rfloor)} a(t) = \surd r.\ \underline{a} \gg \lfloor r+1\rfloor = \underline{a}.$$

4.8 PROPOSITION: $\lceil \underline{a}\cdot x \rceil^{n+1}(y) = \underline{a}\cdot\lceil x \rceil^{n+1}(y)$.

PROOF: $\lceil \underline{a}\cdot x \rceil^{n+1}(y) = \sqrt{r}.\lceil r \gg \underline{a}\cdot x \rceil_\rho^{\lfloor r+n+1 \rfloor}(\lfloor r+n+1 \rfloor \gg y) =$

$= \sqrt{r}.\lceil \int_{t\in(r,\lfloor r+1 \rfloor)} a(t)\cdot x \rceil_\rho^{\lfloor r+n+1 \rfloor}(\lfloor r+n+1 \rfloor \gg y) =$

$= \sqrt{r}. \int_{t\in(r,\lfloor r+1 \rfloor)} a(t)\cdot\lceil t \gg x \rceil_\rho^{\lfloor r+n+1 \rfloor}(\lfloor r+n+1 \rfloor \gg y) =$ (since $\lfloor r \rfloor = \lfloor t \rfloor$)

$= \sqrt{r}. \int_{t\in(r,\lfloor r+1 \rfloor)} a(t)\cdot\lceil t \gg x \rceil_\rho^{\lfloor t+n+1 \rfloor}(\lfloor t+n+1 \rfloor \gg y) =$

$= \sqrt{r}. \int_{t\in(r,\lfloor r+1 \rfloor)} a(t)\cdot \sqrt{s}.\lceil s \gg x \rceil_\rho^{\lfloor s+n+1 \rfloor}(\lfloor s+n+1 \rfloor \gg y) =$

$= \sqrt{r}. \int_{t\in(r,\lfloor r+1 \rfloor)} a(t)\cdot\lceil x \rceil^{n+1}(y) = \sqrt{r}.\left(\int_{t\in(r,\lfloor r+1 \rfloor)} a(t)\right)\cdot\lceil x \rceil^{n+1}(y) = \left(\sqrt{r}. \int_{t\in(r,\lfloor r+1 \rfloor)} a(t)\right)\cdot\lceil x \rceil^{n+1}(y) =$

$= \underline{a}\cdot\lceil x \rceil^{n+1}(y)$.

4.9 PROPOSITION: $\lceil x + y \rceil^{n+1}(z) = \lceil x \rceil^{n+1}(z) + \lceil y \rceil^{n+1}(z)$.

PROOF: Immediate.

4.10 PROPOSITION: $\lceil \sigma_d(x) \rceil^{n+1}(y) = \sigma_d(\lceil x \rceil^n(y))$.

PROOF: $\lceil \sigma_d(x) \rceil^{n+1}(y) = \sqrt{r}.\lceil r \gg \sigma_d(x) \rceil_\rho^{\lfloor r+n+1 \rfloor}(\lfloor r+n+1 \rfloor \gg y) =$

$= \sqrt{r}.\lceil r \gg (\sqrt{t}.\lfloor t+1 \rfloor \gg x) \rceil_\rho^{\lfloor r+n+1 \rfloor}(\lfloor r+n+1 \rfloor \gg y) = \sqrt{r}.\lceil \lfloor r+1 \rfloor \gg x \rceil_\rho^{\lfloor r+n+1 \rfloor}(\lfloor r+n+1 \rfloor \gg y) =$

$\sqrt{r}.\lfloor r+1 \rfloor \gg \lceil x \rceil^n(y) = \sigma_d(\lceil x \rceil^n(y))$.

4.11 PROPOSITION: $\lceil \lfloor x \rfloor(y) \rceil^{n+1}(z) = \lfloor \lceil x \rceil^{n+1}(z) \rfloor(\lceil y \rceil^n(z))$.

PROOF: $\lceil \lfloor x \rfloor(y) \rceil^{n+1}(z) = \lceil x \gg_d 1 + \sigma_d(y) \rceil^{n+1}(z) = \lceil x \gg_d 1 \rceil^{n+1}(z) + \lceil \sigma_d(y) \rceil^{n+1}(z) =^*$

$= \lceil x \rceil^{n+1}(z) \gg_d 1 + \sigma_d(\lceil y \rceil^n(z)) = \lfloor \lceil x \rceil^{n+1}(z) \rfloor(\lceil y \rceil^n(z))$.

The second half of the equality * is explained by 4.10, the first half is based on the following identity:

$s \le t \Rightarrow \lceil x \gg s \rceil_\rho^t(y) = \lceil x \rceil_\rho^t(y) \gg s$.

This identity can be proved for all closed terms by structural induction.

4.12 LIVENESS.

We have that the rigid processes in M_A^* correspond to the real time processes involving absolute time notation only. We can also define a notion that gives the processes that can be specified using relative time notation only by means of the following definition.

We define the predicate $\mathcal{L}$: $M_A^* \to BOOL^*$ as follows:

$$\mathcal{L}(X) = \lambda t.\, t \gg X \neq \delta(t).$$

We call a process X *live* if $\mathcal{L}(X) = \lambda t.$ true. We obtain the following properties:

1. $\mathcal{L}(a[0]) = \mathcal{L}(\delta[0]) = \mathcal{L}(a(0)) = \mathcal{L}(\delta(0)) = \lambda t.$ false
2. $\mathcal{L}(a[n+1]) = \mathcal{L}(\delta[n+1]) = \lambda t.$ true

3. $\mathcal{L}(a(n+1)) = \mathcal{L}(\delta(n+1)) = \lambda t.\ t < n+1$
4. $\mathcal{L}(a(r)) = \mathcal{L}(\delta(r)) = \lambda t.\ t < r$
5. $\mathcal{L}(x + y) = \lambda t.\ \mathcal{L}(x)(t) \vee \mathcal{L}(y)(t)x$
6. $\mathcal{L}(x \cdot y) = \mathcal{L}(x)$
7. $\mathcal{L}(x \between y) = \mathcal{L}(x \between\!\!\!\between y) = \mathcal{L}(x \mid y) = \lambda t.\ \mathcal{L}(x)(t) \wedge \mathcal{L}(y)(t)$
8. $\mathcal{L}(\int_{v \in V} X) = \lambda t.\ \mathbb{W}_{v \in V} \mathcal{L}(X)(t)$.

5. DISCRETISATION.

In the previous sections, we have sketched a theory ACPρ√I and a model M_A^* that contain both real time and discrete time processes. Now we will concentrate on the reduct of the theory and the subalgebra of the model that contain only discrete time processes. For this reason, we introduce the discretisation operator $\mathcal{D}$.

5.1 DEFINITION.

First of all, we define discretisation on P^*. We have the axioms in table 11.

$\mathcal{D}(a(t)) = a(\lfloor t+1 \rfloor)$
$\mathcal{D}(x + y) = \mathcal{D}(x) + \mathcal{D}(y)$
$\mathcal{D}(x \cdot y) = \mathcal{D}(x) \cdot \mathcal{D}(y)$
$\mathcal{D}(\int_{v \in V} X) = \int_{v \in V} \mathcal{D}(X)$
$\mathcal{D}(\sqrt{t}.\ X) = \sqrt{t}.\ \mathcal{D}(X)$

TABLE 11. Discretisation.

5.2 PROPERTIES.

1. $\mathcal{D}$ does not distribute over $\between$, $\between\!\!\!\between$, $\mid$.
2. $\mathcal{D}(\partial_H(x)) = \partial_H(\mathcal{D}(x))$
3. $\lfloor U(x) \rfloor \leq U(\mathcal{D}(x)) \leq \lfloor U(x) + 1 \rfloor$.

In 3, both extremes are possible. For, taking $x = \int_{t \in (1,2)} a(t)$ gives $U(\mathcal{D}(x)) = U(\mathcal{D}(\int_{t \in (1,2)} a(t))) =$

$= U(\int_{t \in (1,2)} a(2)) = U(a(2)) = U(\int_{t \in (1,2)} a(t)) = U(x) = 2 = \lfloor U(x) \rfloor$, while on the other hand taking

$x = a(2)$ gives $U(\mathcal{D}(x)) = U(a(3)) = 3 = \lfloor 2 + 1 \rfloor = \lfloor U(a(2)) + 1 \rfloor = \lfloor U(x) + 1 \rfloor$.

5.3 DEFINITION.

We say that a process x is *discretised*, $\mathcal{DIS}(x)$, if there is a process $y \in P^*$ such that $\mathcal{D}(y) = x$. Also, we define this concept on the model M_A^*. Let $f \in M_A^*$. Then f is discretised, $f \in \mathcal{DIS}M_A^*$, if:

i. for all $t \geq 0$, all actions of f(t) have a timestamp not in $\mathbb{N}$.

ii. for all $t \geq 0$ we have the following: let σ be a sequence of actions in $f(t)$ of the form $a_1(r_1)\ a_2(r_2) \ldots a_n(r_n)$ with $r_1 < r_2 < \ldots < r_n < r$. Let in $f(t)$ the sequence of actions σ from the root be followed by the step $p \xrightarrow{a(r)} q$. Then:

a. for each r' with $r < r' < \lfloor r+1 \rfloor$ there is a q' with $p \xrightarrow{a(r')} q'$ and $q' \leftrightarrow r' \gg q$.

b. for each r' with $\max(r_n, \lfloor r \rfloor) < r' < r$ there is a q' with $p \xrightarrow{a(r')} q'$ and $q \leftrightarrow r \gg q'$.

5.4 EXAMPLES.

1. $\mathcal{DIS}(a(n))$, $\mathcal{DIS}(a[n])$, $\mathcal{DIS}(\delta(n))$, $\mathcal{DIS}(\delta[n])$, $\neg\mathcal{DIS}(a(t))$.
2. $\mathcal{DIS}(x) \Rightarrow \mathcal{DIS}(\partial_H(x))$, $\mathcal{DIS}(\sigma_d(x))$, $\mathcal{DIS}(1 \gg_d x)$, $\mathcal{DIS}(x \gg_d 1)$.
3. $\mathcal{DIS}(x) \wedge \mathcal{DIS}(y) \Rightarrow \mathcal{DIS}(x + y)$, $\mathcal{DIS}(x \cdot y)$, $\mathcal{DIS}(x \| y)$, $\mathcal{DIS}(x \between y)$, $\mathcal{DIS}(x \mid y)$.
4. for all $v \in V$ $\mathcal{DIS}(X) \Rightarrow \mathcal{DIS}(\int_{v \in V} X)$.
5. $\mathcal{DIS}(x) \wedge \neg D(x) \Rightarrow x = x \gg_d 1$.
6. $\mathcal{DIS}(x) \wedge D(x) \Rightarrow x = x \gg_d 1 + \sigma_d(1 \gg_d x)$.
7. $\mathcal{DIS}(x) \Rightarrow \mathcal{L}(x)$
8. $\mathcal{DIS}(x) \Rightarrow \underline{a} \between x = \underline{a} \cdot x$
9. $\mathcal{DIS}(x) \wedge \mathcal{DIS}(y) \Rightarrow \underline{a} \cdot x \between y = \underline{a} \cdot (x \| y)$
10. $\mathcal{DIS}(x) \wedge \mathcal{DIS}(y) \wedge D(y) \Rightarrow \sigma_d(x) \between y = \sigma_d(x \between 1 \gg_d y)$.

The remarkable point is that $\mathcal{L}(\underline{\delta})$. Thus, the discretisation of δ is live. The discretisation of δ is certainly not equal to $\delta = \delta(0)$.

5.5 AN UNTIMED GRAPH MODEL.

We can define a model that is conceptually much simpler than $\mathcal{DIS}M_A^*$, but contains an isomorphic copy of it. This is the model $\mathbb{G}/\leftrightarrow$ that we will use in the following chapter.

We define a set of process graphs as in [BAW90] with labels from $A \cup \{\sigma\}$ satisfying two extra conditions:

i. every node has *at most one* outgoing σ-labeled edge;

ii. a σ-labeled edge may not lead to a termination node.

Let $\mathbb{G}$ be the set of such process graphs with cardinality $\leq 2^{\aleph_0}$. To state this precisely, an element of $\mathbb{G}$ is a quadruple $\langle N, E, r, T \rangle$ where N is the set of *nodes*, $E \subseteq N \times A \cup \{\sigma\} \times N$ is the set of *edges*, $r \in N$ is the *root node*, and $T \subseteq N$ is the set of *termination nodes*. We will always have that a termination node has no outgoing edges. A node without outgoing edges that is not a termination node is called a *deadlock node*.

We define a mapping from $\mathcal{DIS}M_A^*$ to $\mathbb{G}/\leftrightarrow$ by defining a mapping ϕ from real time transition systems to process graphs. Let R be a real time transition system. For simplicity, we assume that R is actually a tree, so each node has at most one incoming transition. If $\langle r, 0 \rangle$ is the root of R ($r \in S$), we take r as the root of $\phi(R)$. The set of states of $\phi(R)$ is S, the set of states of R. Suppose $\langle r, 0 \rangle \xrightarrow{a(t)} \langle s, t \rangle$ is a transition in R from the root. In $\phi(R)$, we take from the root then $\lfloor t \rfloor$ σ-transitions followed by an a-transition to state s. In case of several transitions in R from the root, we share σ-transitions in $\phi(R)$ as much as possible. We proceed likewise for the other states: if $\langle s, u \rangle \xrightarrow{a(t)} \langle s', t \rangle$ is a transition in R,

we take from s in $\phi(R)$ $\lfloor t \rfloor - \lfloor u \rfloor$ σ-transitions followed by an a-transition to s'. This describes f. We then obtain a mapping from $\mathcal{DIS}\mathbb{M}_A^*$ to $\mathbb{G}/\leftrightarrow$ by mapping a representative of $f \in \mathcal{DIS}\mathbb{M}_A^*$ to $R(f(0))/\leftrightarrow$.

The inverse mapping can be defined along the same lines. We identify the domain of $\mathbb{G}$ with some subset of S, the fixed domain of states that was mentioned in 2.13. If r is the root of $g \in \mathbb{G}$, and $r \xrightarrow{\sigma} \dots \xrightarrow{\sigma} \xrightarrow{a} s$ a sequence of transitions with n σ-steps, add transitions $\langle r, 0\rangle \xrightarrow{a(t)} \langle s, t\rangle$ for each t between n and n+1. Next, if node s of g corresponds to node $\langle s, u\rangle$, and $s \xrightarrow{a} s'$ is a transition in g, add transitions $\langle s, u\rangle \xrightarrow{a(t)} \langle s', t\rangle$ for each t between u and $\lfloor u+1 \rfloor$. On the other hand, if $s \xrightarrow{\sigma} \dots \xrightarrow{\sigma} \xrightarrow{a} s'$ is a sequence of transitions in g with n σ-steps, $n \geq 1$, add transitions $\langle s, u\rangle \xrightarrow{a(t)} \langle s', t\rangle$ for each t between $\lfloor u+n \rfloor$ and $\lfloor u+n+1 \rfloor$.

We leave the verification that this indeed defines an isomorphism to the reader.

6. DISCRETE TIME PROCESS ALGEBRA.

Now we will incrementally define axiomatisations for discrete time subalgebras of our theory.

6.1 BPA_{dt}.

We start from:

- a unary operator for (unit) delay like in TCCS [MOT89];
- time factorisation as in ATP [NIS90], TCCS [MOT89], TPCCS [HAN91] and TPL [HER90];
- the interpretation of + as the weak choice of TCCS;
- a significant difference with TPL [HER90] because visible actions cannot idle (here, we follow TCCS, ACP_t and ATP).

The signature of BPA_{dt} is as follows:

•	constants	$\underline{a}$	a in the current time slice ($a \in A$).
•	functions	$+: P \times P \to P$	alternative composition
		$\cdot: P \times P \to P$	sequential composition
		$\sigma_d: P \to P$	discrete time unit delay

The axioms of BPA_{dt} are shown in table 12. Axioms DT1,2 appear in [BAB91b]. DT1 is proved in 4.2, DT2 in 4.3.

$x + y = y + x$	A1
$(x + y) + z = x + (y + z)$	A2
$x + x = x$	A3
$(x + y)\cdot z = x\cdot z + y\cdot z$	A4
$(x\cdot y)\cdot z = x\cdot(y\cdot z)$	A5
$\sigma_d(x) + \sigma_d(y) = \sigma_d(x + y)$	DT1
$\sigma_d(x)\cdot y = \sigma_d(x\cdot y)$	DT2

TABLE 12. BPA_{dt}.

6.2 PROCESS GRAPHS.

The semantic domain for BPA_{dt} consists of process graphs as defined in 5.5. We will map every closed term to a graph in $\mathbb{G}$ by using a structured operational semantics.

6.3 STRUCTURED OPERATIONAL SEMANTICS.

We can give a transition system specification for BPA_{dt} by means of the following rules ($a \in A$).

$$\underline{a \xrightarrow{a} \surd}$$

$$\frac{x \xrightarrow{a} x'}{x \cdot y \xrightarrow{a} x' \cdot y} \qquad \frac{x \xrightarrow{a} \surd}{x \cdot y \xrightarrow{a} y}$$

$$\frac{x \xrightarrow{a} x'}{x+y \xrightarrow{a} x',\ y+x \xrightarrow{a} x'} \qquad \frac{x \xrightarrow{a} \surd}{x+y \xrightarrow{a} \surd,\ y+x \xrightarrow{a} \surd}$$

$$\sigma_d(x) \xrightarrow{\sigma} x \qquad \frac{x \xrightarrow{\sigma} x'}{x \cdot y \xrightarrow{\sigma} x' \cdot y}$$

$$\frac{x \xrightarrow{\sigma} x',\ y \xrightarrow{\sigma} y'}{x+y \xrightarrow{\sigma} x'+y'} \qquad \frac{x \xrightarrow{\sigma} x',\ y \not\xrightarrow{\sigma}}{x+y \xrightarrow{\sigma} x',\ y+x \xrightarrow{\sigma} x'}$$

TABLE 13. Operational semantics of BPA_{dt}.

Now we want to show that these rules give rise to a unique transition relation on closed terms. In order to do this, we use results and terminology of [GRO90b]. A *strict stratification* of a transition system specification is a mapping S from closed transitions to ordinal numbers such that for all possible substitutions of the variables in a rule, the mapping applied to the conclusion is yields a larger ordinal than the mapping applied to all instances of all premises. In this case, it is easy to give a strict stratification: put $S(t \xrightarrow{a} t') = n$ where n is the number of function names in t. Theorem 4.2.18 in [GRO90b] now shows that the transition system specification in table 13 determines a unique transition relation on closed terms. It is easily verified that the process graphs generated by this transition relation satisfy the conditions of $\mathbb{G}$.

Bisimulation is defined as usual, so a symmetric binary relation R on process terms is a bisimulation iff the following transfer conditions hold:

i. if $R(p,q)$ and $p \xrightarrow{u} p'$ ($u \in A \cup \{\sigma\}$), then there is q' such that $q \xrightarrow{u} q'$ and $R(s',t')$;

ii. if $R(p,q)$, then $p \xrightarrow{a} \surd$ ($a \in A$) iff $q \xrightarrow{a} \surd$.

Two terms p,q are bisimilar, $p \leftrightarroweq q$, if there exists a bisimulation relating them. We want to make the set of process terms modulo bisimulation into a model for BPA_{dt}. In order to do this, we first need to

know that bisimulation is a congruence. We prove this again using the theory developed in [GRO90b]. We first need some definitions.

So, consider a transition system specification with a rule r. We say this rule is in *ntyft-format* if it is of the form

$$\frac{\{t_k \xrightarrow{a_k} y_k \mid k \in K\} \cup \{t_l \xrightarrow{a_l} \not\rightarrow \mid l \in L\}}{f(x_1,\ldots,x_n) \xrightarrow{a} t}$$

where K,L are index sets, y_k,x_i are all distinct variables, a_k,b_l,a are labels, f a function symbol of arity n, and t_k,t_l,t are arbitrary (open) terms. We say a rule is in *ntyxt-format* if it is of the form

$$\frac{\{t_k \xrightarrow{a_k} y_k \mid k \in K\} \cup \{t_l \xrightarrow{a_l} \not\rightarrow \mid l \in L\}}{x \xrightarrow{a} t}$$

with similar conventions. We say that a transition system specification is in *ntyft/ntyxt-format* if all its rules are in *ntyft* or *ntyxt* format. Now theorem 4.4.12 of [GRO90b] says that for any stratifiable transition system specification with finite premise sets, bisimulation is a congruence.

Unfortunately, the transition system specification in table 13 is not in *ntyft/ntyxt* format. However, it is easy to define an equivalent specification that is. To do that, we double the set of labels to $A \cup \{a\surd : a \in A\}$, replace the axiom $\underline{a} \xrightarrow{a} \surd$ by $\underline{a} \xrightarrow{a\surd} \underline{\delta}$, and replace each transition of the form $x \xrightarrow{a} \surd$ by a transition $x \xrightarrow{a\surd} \underline{\delta}$. It is easy to see that the resulting set of rules is in *ntyft/ntyxt* format. We leave it to the reader to show that the two specifications are equivalent.

Next, we will show that the axiomatisation of BPA_{dt} is sound and complete for the set of transition graphs modulo bisimulation. In order to do this, we first need the notion of a basic term. This is a term that corresponds more directly to its transition graph.

6.4 BASIC TERMS.

In order to define the set of basic terms inductively, we need the auxiliary notion of an A-basic term (a term with no leading σ_d). We define both notions simultaneously.

1. every A-basic term is a basic term;
2. if $a \in A$, then $\underline{a}$ is an A-basic term;
3. if $a \in A$ and t is a basic term, then $\underline{a}\cdot t$ is an A-basic term;
4. if t is a basic term, then $\sigma_d(t)$ is a basic term;
5. if t,s are A-basic terms, then $t+s$ is an A-basic term;
6. if t is an A-basic term, and s is a basic term, then $t + \sigma_d(s)$ is a basic term.

Some reflection leads to the insight that the basic terms are exactly the terms of one of the following two forms:

a. $\sum_{i<n} \underline{a_i}\cdot t_i + \sum_{j<m} \underline{b_j}$, with $n+m>0$, $a_i,b_j \in A$ and t_i basic;

b. $\sum_{i<n} \underline{a_i}\cdot t_i + \sum_{j<m} \underline{b_j} + \sigma_d(t)$, with $n,m\geq 0$, $a_i,b_j \in A$ and t,t_i basic.

It is easy to see that every basic term can be mapped to an element of $\mathbb{G}$ in a straightforward way.

6.5 NORMALISATION.

Let t be a closed BPA_{dt}-term. Then there is a basic term s such that $BPA_{dt} \vdash t = s$.

PROOF. Similarly to what is done in [BAW90], we can prove this fact by considering axioms A4, A5, DT1 and DT2 as rewrite rules from left to right. A normal form of a closed term will be a basic term.

6.6 THEOREM.

Let p,q be closed BPA_{dt}-expressions. Then we have $BPA_{dt} \vdash p=q \Leftrightarrow \mathbb{G} \models p \leftrightarrows q$,
i.e. BPA_{dt} is sound and complete for the set of transition graphs modulo bisimulation.

PROOF. Soundness is not difficult, completeness requires an argument. Let BPA(A) be the theory of Basic Process Algebra (without operator σ_d, with only laws A1-A5) over set of atomic actions A. Similarly, we write $BPA_{dt}(A)$. Now we map $BPA_{dt}(A)$-terms to $BPA(A \cup \{\sigma\})$-terms as follows:

- $\phi(\underline{a}) = a$
- $\phi(x + y) = \phi(x) + \phi(y)$
- $\phi(x \cdot y) = \phi(x) \cdot \phi(y)$
- $\phi(\sigma_d(x)) = \sigma \cdot \phi(x)$.

Now let bisimulating closed terms p,q be given. By 2.6, we can assume that p,q are basic terms. Since p,q are basic, it follows that $\phi(p) \leftrightarrows \phi(q)$. But then, by completeness of BPA for strong bisimulation (see [BAW90]), it follows that $BPA(A \cup \{\sigma\}) \vdash \phi(p) = \phi(q)$. We can mimic each step in this proof by a step in BPA_{dt} (using the inverse mapping of ϕ), and thereby it follows that $BPA_{dt}(A) \vdash p=q$.

6.7 GRAPH MODEL.

Thus, we have found a sound and complete model for BPA_{dt}. It is also possible to define this model directly, not by giving a transition system specification, but by defining an interpretation of the constants and operators on process graphs. For the purposes of extensions with the silent step further on, it is better to work with *process trees* only, i.e. graphs where each node has only one incoming edge (and the root has no incoming edge).

1. The constant $\underline{a}$ is mapped to the process tree with two nodes, the first the root, the second a termination node, with one edge labeled a from the root node to the termination node.
2. Given trees g,h, the tree g+h is formed by identifying the roots of g and h. Next, if both roots have an outgoing σ-edge, both these edges are removed, and a new σ-edge is added to the sum of the trees the original σ-edges where going to. If necessary, the procedure is repeated.
3. Given trees g,h, the tree $g \cdot h$ is formed by appending a copy of h to each termination node of g. The root of g is the root of $g \cdot h$, and only the termination nodes of h are termination nodes of $g \cdot h$.
4. Given tree g, the tree $\sigma_d(g)$ is formed by adding a new root node, and an edge labeled σ from the new root to the old root.

6.8 INACTION.

$BPA_{\delta dt}$ is obtained by introducing $\underline{\delta}$ as a constant representing inaction (time stop at the end of the current time slice). The axioms for $\underline{\delta}$ are standard (table 14).

$x + \underline{\delta} = x$	A6
$\underline{\delta} \cdot x = \underline{\delta}$	A7

TABLE 14. Additional axioms of $BPA_{\delta dt}$.

The operational meaning of $\underline{\delta}$ is a process that allows no step whatsoever. In the graph model, $\underline{\delta}$ will be modeled by the tree with no edges, and one node which is the root but not a termination node. We can again define basic terms over the extended theory, by adding one clause to definition 6.4:
7. $\underline{\delta}$ is a basic term.
Then, normalisation goes through as before.

6.9 MOTIVATION.
$\underline{\delta}$ is the same as 0 of TCCS, and it seems to be equivalent to an infinite τ-loop in TPL. $\underline{\delta}$ is like δ in [GRO90a] and like nil in [GM91]. $\underline{\delta}$ is a so-called *time stop*.

Although time stops do not occur in actual computing, we see no objection to their presence in the calculus. So we so not follow ATP's design rationale, which excludes time stops on a priori grounds. We notice that TPL has no constant that introduces a time stop.

Next, we add parallel composition. We start by describing a system with merge without communication, a so-called free merge.

6.10 $PA_{\delta dt}$.
We start by adding to the signature of $BPA_{\delta dt}$ some extra operators and an auxiliary sort.

$\neg true = false$	BOOL1
$\neg false = true$	BOOL2
$true \vee b = true$	BOOL3
$false \vee b = b$	BOOL4
$x \lhd true \rhd y = x$	CO1
$x \lhd false \rhd y = y$	CO2
$1 \gg_d \underline{a} = \underline{\delta}$	UTB1
$1 \gg_d (x + y) = 1 \gg_d x + 1 \gg_d y$	UTB2
$1 \gg_d (x \cdot y) = (1 \gg_d x) \cdot y$	UTB3
$1 \gg_d \sigma_d(x) = x$	UTB4
$D(\underline{a}) = false$	DEL1
$D(x + y) = D(x) \vee D(y)$	DEL2
$D(x \cdot y) = D(x)$	DEL3
$D(\sigma_d(x)) = true$	DEL4

TABLE 15. Axioms for additional operators.

•	sort	BOOL	booleans
•	constants	$\mathsf{true}: \mathsf{B}$	true
		$\mathsf{false}: \mathsf{B}$	false
•	functions	$\neg: \mathsf{BOOL} \to \mathsf{BOOL}$	negation
		$\vee: \mathsf{BOOL} \times \mathsf{BOOL} \to \mathsf{BOOL}$	disjunction
		$.\lhd.\rhd.: \mathsf{BOOL} \times \mathsf{P} \times \mathsf{BOOL} \to \mathsf{P}$	conditional operator (if then else)
		$1 \gg_d: \mathsf{P} \to \mathsf{P}$	initialisation in the following time slice
		$\mathsf{D}: \mathsf{P} \to \mathsf{B}$	delayability.

Axioms for this additional syntax are shown in table 15 ($b \in \mathsf{BOOL}$, $a \in A_\delta$, $x,y \in \mathsf{P}$). They already appear in [BAB91b].

Then, the merge with its auxiliary operator left merge is introduced. Syntax:

•	functions	$\|: \mathsf{P} \times \mathsf{P} \to \mathsf{P}$	merge
		$\lfloor: \mathsf{P} \times \mathsf{P} \to \mathsf{P}$	left merge.

Axioms are in table 16 ($a \in A_\delta$, $x,y,z \in \mathsf{P}$). Axiom DTM1 is from [BAB91b].

$x \| y = x \lfloor y + y \lfloor x$	M1
$\underline{a} \lfloor x = \underline{a} \cdot x$	M2
$(\underline{a} \cdot x) \lfloor y = \underline{a} \cdot (x \| y)$	M3
$(x + y) \lfloor z = x \lfloor z + y \lfloor z$	M4
$\sigma_d(x) \lfloor y = \sigma_d(x \lfloor (1 \gg_d y)) \lhd \mathsf{D}(y) \rhd \underline{\delta}$	DTM1

TABLE 16. Remaining axioms of $\mathrm{PA}_{\delta dt}$.

Operational rules for the additional process constructors are shown in table 17.

$$\frac{x \xrightarrow{a} x'}{x \| y \xrightarrow{a} x' \| y,\ y \| x \xrightarrow{a} y \| x',\ x \lfloor y \xrightarrow{a} x' \| y}$$

$$\frac{x \xrightarrow{a} \surd}{x \| y \xrightarrow{a} y,\ y \| x \xrightarrow{a} y,\ x \lfloor y \xrightarrow{a} y}$$

$$\frac{x \xrightarrow{\sigma} x',\ y \xrightarrow{\sigma} y'}{x \| y \xrightarrow{\sigma} x' \| y',\ x \lfloor y \xrightarrow{\sigma} x' \lfloor y'}$$

$$\frac{x \xrightarrow{\sigma} y,\ y \xrightarrow{a} z}{1 \gg_d x \xrightarrow{a} z} \qquad \frac{x \xrightarrow{\sigma} y,\ y \xrightarrow{a} \surd}{1 \gg_d x \xrightarrow{a} \surd}$$

$$\frac{x \xrightarrow{\sigma} y,\ y \xrightarrow{\sigma} z}{1 \gg_d x \xrightarrow{\sigma} z}$$

TABLE 17. Operational semantics of $\mathrm{PA}_{\delta dt}$.

As before, these operational rules can easily be transformed into *ntyft/ntyxt* format, so that bisimulation is a congruence.

6.11 NORMALISATION.
By using term rewrite analysis, we can prove that each closed term is equal to a basic term over $BPA_{\delta dt}$. From this, the completeness of the axioms for finite processes can be deduced.

6.12 MOTIVATION.
Our definition of merge is similar to the definitions in TCCS, ATP, ACP_t and TPL (the correspondence with the TPL definition holds in the absence of communication; in the presence of communication TPL's merge will give priority to internal communications). So it follows that without communication the relation between merge and discrete time is not controversial.

We depart from ATP by allowing time stops. Indeed, in our setting $\sigma_d(\underline{a}\cdot\underline{b}) \| \underline{\delta}$ is unable to perform a σ-step. This is a matter of taste. The presence of time stops seems not to spoil the algebra at all.

6.13 COMMUNICATION.
If we add communication, we have a given *communication function* $| : A_\delta \times A_\delta \to A_\delta$ as an extra parameter of the theory. This communication function is required to be commutative, associative and has δ as a neutral element (axioms C1,2,3 of table 6 in 2.3). Furthermore, in order to axiomatise the merge, we have an additional auxiliary operator $|$ (communication merge). We replace the axioms in table 16 by the axioms in table 18 below ($a,b,c \in A_\delta$, $x,y,z \in P$). The DTM axioms are from [BAB91b].

- functions $| : P \times P \to P$ communication merge

$\underline{a} \mid \underline{b} = \underline{c}$	if $a \mid b = c$	C4
$x \| y = x \mathbin{\mathbb{L}} y + y \mathbin{\mathbb{L}} x + x \mid y$		CM1
$\underline{a} \mathbin{\mathbb{L}} x = \underline{a}\cdot x$		CM2
$(\underline{a}\cdot x) \mathbin{\mathbb{L}} y = \underline{a}\cdot(x \| y)$		CM3
$(x + y) \mathbin{\mathbb{L}} z = x \mathbin{\mathbb{L}} z + y \mathbin{\mathbb{L}} z$		CM4
$\sigma_d(x) \mathbin{\mathbb{L}} y = \sigma_d(x \mathbin{\mathbb{L}} (1 \gg_d y)) \lhd D(y) \rhd \underline{\delta}$		DTM1
$(\underline{a}\cdot x) \mid \underline{b} = (\underline{a} \mid \underline{b})\cdot x$		CM5
$\underline{a} \mid (\underline{b}\cdot x) = (\underline{a} \mid \underline{b})\cdot x$		CM6
$(\underline{a}\cdot x) \mid (\underline{b}\cdot y) = (\underline{a} \mid \underline{b})\cdot(x \| y)$		CM7
$(x+y) \mid z = x \mid z + y \mid z$		CM8
$x \mid (y+z) = x \mid y + x \mid z$		CM9
$\underline{a} \mid \sigma_d(x) = \underline{\delta}$		DTM2
$\sigma_d(x) \mid \underline{a} = \underline{\delta}$		DTM3
$(\underline{a}\cdot x) \mid \sigma_d(y) = \underline{\delta}$		DTM4
$\sigma_d(x) \mid (\underline{a}\cdot y) = \underline{\delta}$		DTM5
$\sigma_d(x) \mid \sigma_d(y) = \sigma_d(x \mid y)$		DTM6

TABLE 18. Merge in ACP_{dt}.

Encapsulation ∂_H: $P \to P$ for each $H \subseteq A$ has the same equations as in ACP plus the additional axiom DTM7.

$\partial_H(\underline{a}) = \underline{a}$	if $a \notin H$	D1
$\partial_H(\underline{a}) = \underline{\delta}$	if $a \in H$	D2
$\partial_H(x + y) = \partial_H(x) + \partial_H(y)$		D3
$\partial_H(x \cdot y) = \partial_H(x) \cdot \partial_H(y)$		D4
$\partial_H(\sigma_d(x)) = \sigma_d(\partial_H(x))$		DTM7

TABLE 19. Remaining axioms of ACP_{dt}.

6.14 OPERATIONAL SEMANTICS.
We have to add several operational rules to those of $PA_{\delta dt}$ ($a,b,c \in A$).

$$\frac{x \xrightarrow{a} x',\ y \xrightarrow{b} y',\ a \mid b = c}{x \| y \xrightarrow{c} x' \| y',\ x \mid y \xrightarrow{c} x' \| y'}$$

$$\frac{x \xrightarrow{a} x',\ y \xrightarrow{b} \surd,\ a \mid b = c}{x \| y \xrightarrow{c} x',\ x \mid y \xrightarrow{c} x',\ y \| x \xrightarrow{c} x',\ y \mid x \xrightarrow{c} x'}$$

$$\frac{x \xrightarrow{a} \surd,\ y \xrightarrow{b} \surd,\ a \mid b = c}{x \| y \xrightarrow{c} \surd,\ x \mid y \xrightarrow{c} \surd}$$

$$\frac{x \xrightarrow{\sigma} x',\ y \xrightarrow{\sigma} y'}{x \mid y \xrightarrow{\sigma} x' \mid y'}$$

$$\frac{x \xrightarrow{a} x',\ a \notin H}{\partial_H(x) \xrightarrow{a} \partial_H(x')} \qquad \frac{x \xrightarrow{a} \surd,\ a \notin H}{\partial_H(x) \xrightarrow{a} \surd} \qquad \frac{x \xrightarrow{\sigma} x'}{\partial_H(x) \xrightarrow{\sigma} \partial_H(x')}$$

TABLE 20. Additional operational rules of ACP_{dt}.

6.15 MOTIVATION.
At this point, some further motivation is needed. The merge of ACP_{dt} works just as the merge of TCCS [MOT89] (taking weak choice of TCCS for +). It is also equivalent to the merge of ATP. It differs from merge in [GRO90a], set up in ACP_t. In fact, ACP_t contains axioms $\sigma \between x = \delta$ and $\sigma x \between y = \delta$. Our main objection against these axioms (which occur in ATP as well) is that they render it impossible to injectively embed ACP_t into ACPρ of [BAB91a].

It follows that the only disagreement in the literature is about the proper semantics of left merge and communication merge. Therefore, our choice for the merge operator itself seems reasonably well motivated.

The syntax of ACP_{dt} is not tailored to practical application. The operators are too low level. We will extend ACP_{dt} with ATP operators in order to extend the expressive power of the language. In addition, we need an auxiliary operator $\_ \gg_d 1$.

6.16 SYNTAX.

• functions	$\gg_d 1 : P \to P$	time out in the current time slice
	$\oplus : P \times P \to P$	strong choice
	$\lfloor . \rfloor(.) : P \times P \to P$	unit delay
	$\lfloor . \rfloor^n(.) : P \times P \to P$	start delay within n ($n \in \mathbb{N}$)
	$\lfloor . \rfloor^\omega : P \to P$	unbounded start delay
	$\lceil . \rceil^n(.) : P \times P \to P$	execution delay.

The last five functions are the Nicollin-Sifakis functions.

6.17 AXIOMS.

Axioms for these auxiliary operators are in table 21. Axioms UTB5-8 and ATP1,2 are taken from [BAB91b]. For ATP1,3,6 we refer to table 9, ATP2 is shown in 4.5, ATP5 in 4.4, ATP7-10 in 4.7-10. UTB8 is shown in 4.6.

$\underline{a} \gg_d 1 = \underline{a}$	UTB5
$(x+y) \gg_d 1 = x \gg_d 1 + y \gg_d 1$	UTB6
$(x \cdot y) \gg_d 1 = (x \gg_d 1) \cdot y$	UTB7
$\sigma_d(x) \gg_d 1 = \underline{\delta}$	UTB8
$\lfloor x \rfloor(y) = x \gg_d 1 + \sigma_d(y)$	ATP1
$x \oplus y = x+y \gg_d 1 +$ $\sigma_d(1 \gg x \oplus 1 \gg y) \lhd D(x) \wedge D(y) \rhd \underline{\delta}$	ATP2
$\lfloor x \rfloor^0(y) = y$	ATP3
$\lfloor x \rfloor^{n+1}(y) = \lfloor x \rfloor(\lfloor x \rfloor^n(y))$	ATP4
$\lfloor x \rfloor^\omega = \lfloor x \rfloor(\lfloor x \rfloor^\omega)$	ATP5
$\lceil x \rceil^0(y) = y$	ATP6
$\lceil \underline{a} \rceil^{n+1}(x) = \underline{a}$	ATP7
$\lceil \underline{a} \cdot x \rceil^{n+1}(y) = \underline{a} \lceil x \rceil^{n+1}(y)$	ATP8
$\lceil x + y \rceil^{n+1}(z) = \lceil x \rceil^{n+1}(z) + \lceil y \rceil^{n+1}(z)$	ATP9
$\lceil \sigma_d(x) \rceil^{n+1}(y) = \sigma_d(\lceil x \rceil^n(y))$	ATP10

TABLE 21. Additional axioms for Nicollin-Sifakis functions.

6.18 OPERATIONAL SEMANTICS.

For the new operators, we have the following operational rules (table 22).

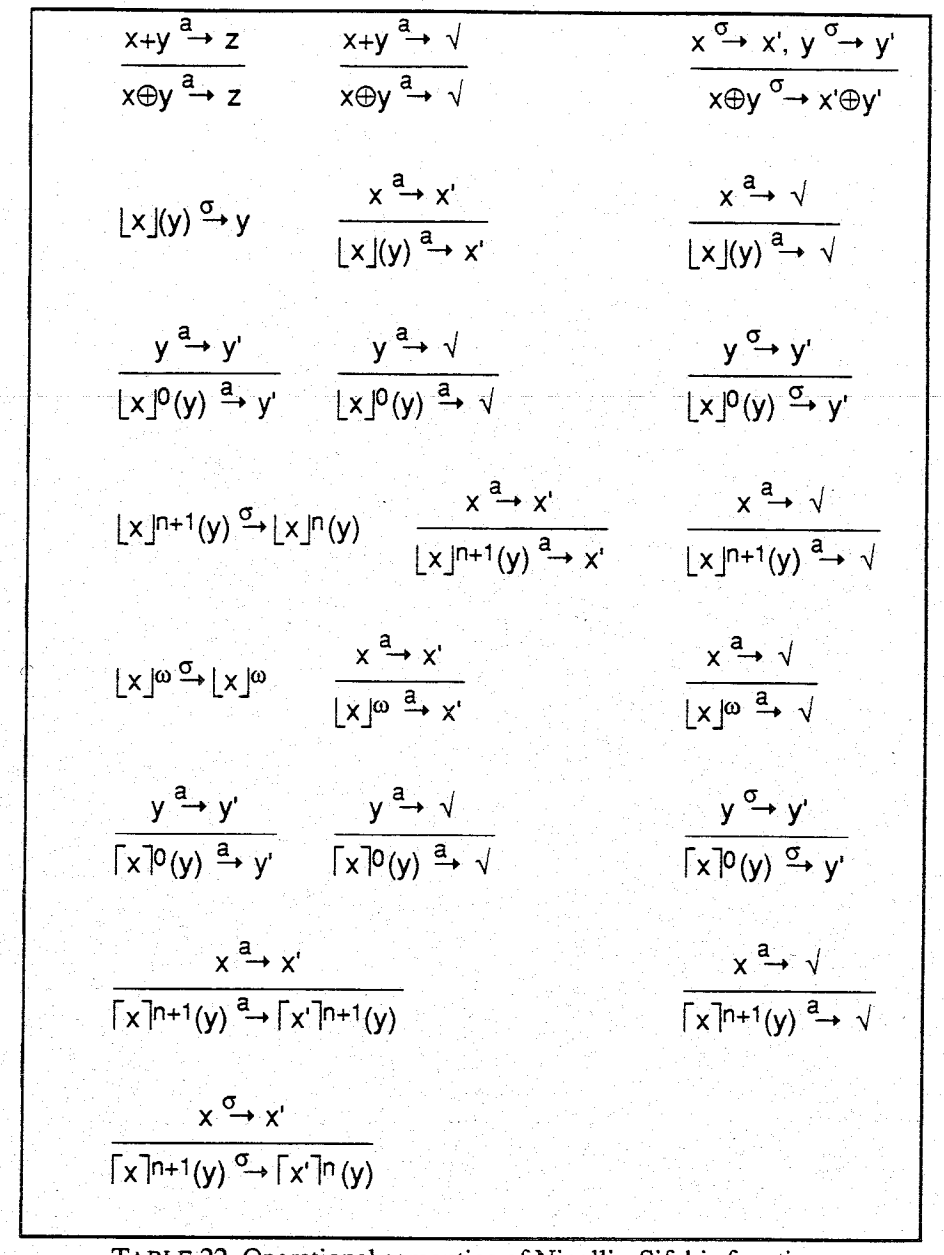


TABLE 22. Operational semantics of Nicollin-Sifakis functions.

6.19. ABSTRACTION.

An advantage of concentrating on the subtheory $ACP_{dt} + ATP$ is that it is easier to add the silent step τ. In real time, this is still under investigation (see [KLU92]), but on our discrete time subtheory, it is relatively straightforward. We will not interpret the constant τ in real time. Clearly, that is an option, but it requires real time abstraction to be worked out. (Besides [KLU92], other approaches exist.)

We add a constant $\underline{\tau}$ ($\tau \notin A$), and for each $I \subseteq A$ an operator $\tau_I: P \to P$. Further, we have an operator τ_σ that substitutes $\underline{\tau}$ for time steps. All axioms in tables 15, 18, 19 are now valid for $a \in A \cup \{\delta, \tau\}$. Additional axioms are shown in table 23 (where also $a \in A \cup \{\delta, \tau\}$). Here, we follow the presentation of the silent step according to *branching bisimulation semantics* (see [BAW90]). Alternatively, we can present an axiomatisation of the silent step in *weak bisimulation semantics* (see [MIL89]). First, we look at branching bisimulation.

$x\underline{\tau} = x$		B1
$x(\underline{\tau}(y + z) + y) = x(y + z)$		B2
$\underline{\tau} \mid \underline{a} = \underline{\delta}$		C4
$\tau_I(\underline{a}) = \underline{a}$	if $a \notin I$	TI1
$\tau_I(\underline{a}) = \underline{\tau}$	if $a \in I$	TI2
$\tau_I(x + y) = \tau_I(x) + \tau_I(y)$		TI3
$\tau_I(x \cdot y) = \tau_I(x) \cdot \tau_I(y)$		TI4
$\tau_I(\sigma_d(x)) = \sigma_d(\tau_I(x))$		TI8
$\tau_\sigma(\underline{a}) = \underline{a}$		TS1
$\tau_\sigma(x + y) = \tau_\sigma(x) + \tau_\sigma(y)$	if $D(x) \vee D(y) = false$	TS2
$\tau_\sigma(x \cdot y) = \tau_\sigma(x) \cdot \tau_\sigma(y)$		TS3
$\tau_\sigma(\sigma_d(x)) = \underline{\tau} \cdot \tau_\sigma(x)$		TS4

TABLE 23. Axioms of ACP^{τ}_{dt}.

6.20 SEMANTICS.

The operational semantics is obtained by using the rules of tables 13, 17, 20 also in case $a = \tau$, but dividing out a different congruence relation. We define branching bisimulation on process graphs as defined in 5.5. For simplicity sake, we assume we are dealing with *process trees* (every graph can be completely unfolded to obtain a tree). Write $s \Rightarrow t$ for nodes in a process tree if either $s=t$ or there is path from s to t (generated by the operational rules) that only contains τ-labels.

A *branching bisimulation* is a binary relation R on nodes of two process trees g,h with the following properties:

i. The roots of g,h are related;

ii. If $R(s,t)$ and $s \xrightarrow{a} s'$ ($a \in A \cup \{\tau\}$), then either $a=\tau$ and $R(s',t)$,or there are t^*, t' in h such that $t \Rightarrow t^* \xrightarrow{a} t'$ and $R(s,t^*)$, $R(s',t')$;

iii. Vice versa: if $R(s,t)$ and $t \xrightarrow{a} t'$ ($a \in A \cup \{\tau\}$), then either $a=\tau$ and $R(s,t')$,or there are s^*, s' in g such that $s \Rightarrow s^* \xrightarrow{a} s'$ and $R(s^*,t)$, $R(s',t')$.

We say that two process trees are *branching bisimilar*, $g \leftrightarrows_b h$, if there is a branching bisimulation between g and h.

Now let R be a branching bisimulation between g,h and s,t nodes in g, resp. h. We say that s,t *satisfy the root condition* if we have:

i. if $R(s,t)$ and $s \xrightarrow{a} s'$ ($a \in A \cup \{\tau\}$), then there is t' in h such that $t \xrightarrow{a} t'$ and $R(s',t')$;

ii. vice versa: if $R(s,t)$ and $t \xrightarrow{a} t'$ ($a \in A \cup \{\tau\}$), then there is s' in g such that $s \xrightarrow{a} s'$ and $R(s',t')$.

Then, we say that two process trees are *rooted branching bisimilar*, $g \leftrightarrows_{rb} h$, if there exists a branching bisimulation R such that:

i. the roots of g,h satisfy the root condition;

ii. whenever $R(s,t)$ and there is a path from the root of g to s with the last transition a σ-transition, then s,t satisfy the root condition;

iii. vice versa.

6.21 FAIR ABSTRACTION.

Besides KFAR and CFAR (see [BAW90]), there is a rule for time:

$\dfrac{x = \sigma_d(x) + y}{\tau\cdot\tau_\sigma(x) = \tau\cdot\tau_\sigma(y)}$	σFAR

TABLE 24. Time abstraction.

Using Nicollin-Sifakis operators, the fair abstraction principle can be formulated as follows:

$$\frac{x = \lfloor y \rfloor(x)}{\tau\cdot\tau_\sigma(x) = \tau\cdot\tau_\sigma(y)} \qquad \sigma\text{FAR},$$

or alternatively:

$$\tau\cdot\tau_\sigma(\lfloor x \rfloor^\omega) = \tau\cdot\tau_\sigma(x) \qquad \sigma\text{FAR}.$$

6.22 WEAK BISIMULATION.

In the weak bisimulation case, not all axioms of tables 15, 18, 19 are valid for τ. Therefore, we have to formulate new axioms for the interaction of the silent step with parallel composition. In weak bisimulation semantics, we replace the axioms B1,2 in table 23 by T1,2,3 below, we replace axiom DEL3 of table 15 by axioms DEL5,6,7 below, we replace axiom UTB3 of table 15 by axioms UTB10,13,14, and we add the axioms TM1,2, TC1,2,3,4, TI0 and TS0 below.

$x\underline{\tau} = x$	T1
$\underline{\tau}x + x = \underline{\tau}x$	T2
$\underline{a}(\underline{\tau}x + y) + y) = \underline{a}(\underline{\tau}x + y) + \underline{a}x$	T3
$D(\underline{a}x) = \text{false}$	DEL5
$D(\underline{\tau}) = \text{false}$	DEL6
$D(\underline{\tau}x) = D(x)$	DEL7
$1 \gg_d \underline{a}x = \underline{\delta}$	UTB10
$1 \gg_d \underline{\tau} = \underline{\delta}$	UTB13
$1 \gg_d \underline{\tau}x = 1 \gg_d x$	UTB14
$\underline{\tau} \between x = \underline{\tau}x$	TM1
$\underline{\tau}x \between y = \underline{\tau}(x \between y)$	TM2
$\underline{\tau} \mid x = \underline{\delta}$	TC1
$x \mid \underline{\tau} = \underline{\delta}$	TC2
$\underline{\tau}x \mid y = x \mid y$	TC3
$x \mid \underline{\tau}y = x \mid y$	TC4
$\tau_I(\underline{\tau}) = \underline{\tau}$	TI0
$\tau_\sigma(\underline{\tau}) = \underline{\tau}$	TS0

TABLE 25. Axioms of $ACP_{\tau dt}$.

For the operational semantics, we can divide out the so-called rooted τ-bisimulation on process trees, using a root condition similar to the one above.

7. EXAMPLES.

7.1 SECURITY COSTS PROTOCOL.

We describe the Security Costs Protocol as it appears in [HER90]. This protocol describes the transition of a message between two locations. Transmission of a message across a secure medium is considered expensive while acknowledgements travel freely. The protocol initially sends the message across an unsecure medium, only resending across the secure medium if an acknowledgement has not arrived before a timeout. We have the following network architecture (fig. 1).

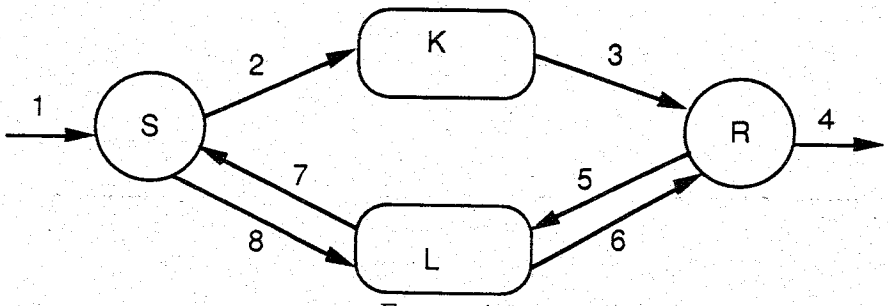


FIGURE 1.

We give recursive specifications for the different components. We use the standard communication function of [BAW90], i.e. the only communications are of the form $r_i(d) \mid s_i(d) = c_i(d)$. In order not to clutter up the notation too much, we leave out the _ signs under the atomic actions.

Sender:

$$S = \sum_{d \in D} \lfloor r1(d) \rfloor^{\omega} \cdot S_d \cdot S$$

$$S_d = s2(d) \cdot \lfloor r7(ack) \cdot r7(ack) \rfloor (s8(d) \cdot r7(ack)) \quad (d \in D).$$

Unreliable medium:

$$K = \sum_{d \in D} \lfloor r2(d) \rfloor^{\omega} \cdot (i \cdot K + i \cdot s3(d) \cdot K) \ .$$

Reliable medium:

$$L = \Big(\lfloor r5(ack) \rfloor^{\omega} \cdot s7(ack) + \sum_{d \in D} \lfloor r8(d) \rfloor^{\omega} \cdot s6(d) \Big) \cdot L.$$

Receiver:

$$R = \Big(\sum_{d \in D} \lfloor r3(d) \rfloor^{\omega} \cdot s5(ack) \cdot s4(d) \cdot s5(ack) + \sum_{d \in D} \lfloor r6(d) \rfloor^{\omega} \cdot s4(d) \cdot s5(ack) \Big) \cdot R.$$

Then the protocol is described by:

$SCP = \tau_{\sigma} \circ \tau_I \circ \partial_H (S \parallel K \parallel L \parallel R)$, where

$H = \{r_i(d), s_i(d) \mid i \in \{2,3,5,6,7,8\}, d \in D\}$, $I = \{c_i(d) \mid i \in \{2,3,5,6,7,8\}, d \in D\} \cup \{i\}$.

Now put $X = \partial_H(S \| K \| L \| R)$, then some straightforward calculations lead to the following recursive specification:

$$X = \sum_{d \in D} \lfloor r1(d) \rfloor^{\omega} \cdot c2(d) \cdot \big(i \cdot \sigma_d(c8(d) \cdot c6(d) \cdot s4(d) \cdot c5(ack) \cdot c7(ack)) +$$

$$+ i \cdot c3(d) \cdot c5(ack) \cdot \big(s4(d) \cdot c7(ack) + c7(ack) \cdot s4(d) \big) \cdot c5(ack) \cdot c7(ack) \big) \cdot X.$$

After abstracting from actions in I, this turns into:

$$\tau_I(X) = \sum_{d \in D} \lfloor r1(d) \rfloor^{\omega} \cdot \big(\tau \cdot \sigma_d(\tau \cdot s4(d)) + \tau \cdot s4(d) \big) \cdot \tau_I(X).$$

Finally, abstracting from time, and using the fair abstraction of 6.21, we obtain:

$$SCP = \tau_\sigma \circ \tau_I(X) = \tau \cdot \sum_{d \in D} r1(d) \cdot s4(d) \cdot SCP.$$

7.2 ALTERNATING BIT PROTOCOL OF [NIS90].
We describe the alternating bit protocol with time out of [NIS90]. Put $B = \{0,1\}$.

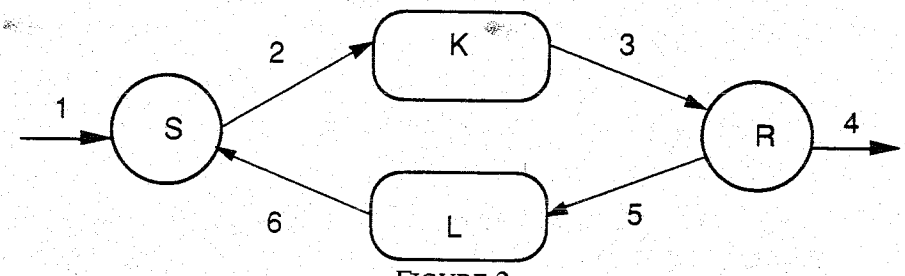


FIGURE 2.

Sender:

$$S = S0 \cdot S1 \cdot S$$

$$Sb = \sum_{d \in D} \lfloor r1(d) \rfloor^{\omega} \cdot Sb_d \qquad (b = 0,1,\ d \in D)$$

$$Sb_d = s2(db) \cdot \lfloor r6(b) + r6(1\text{-}b) \cdot Sb_d \rfloor (Sb_d) \qquad (b = 0,1,\ d \in D).$$

Unreliable data transmission channel:

$$K = \sum_{f \in D \times B} \lfloor r2(f) \rfloor^{\omega} \cdot (i \cdot K + i \cdot s3(f) \cdot K) .$$

Unreliable acknowledgement transmission channel:

$$L = \sum_{b \in B} \lfloor r5(b) \rfloor^{\omega} \cdot (i \cdot L + i \cdot s6(b) \cdot L) .$$

Receiver:

$$R = R1 \cdot R0 \cdot R$$

$$Rb = \sum_{d\in D} \lfloor r3(db) \rfloor^{\omega} \cdot s5(b) \cdot Rb + \sum_{d\in D} \lfloor r3(d(1\text{-}b)) \rfloor^{\omega} \cdot s4(d) \cdot s5(1\text{-}b).$$

Then the protocol is described by:

$ABP = \tau_{\sigma} \circ \tau_I \circ \partial_H(S \| K \| L \| R)$, where

$H = \{ri(x), si(x) \mid i \in \{2,3,5,6\}, x \in \{0,1\} \cup D \times \{0,1\}\}$,

$I = \{ci(x) \mid i \in \{2,3,5,6\}, x \in \{0,1\} \cup D \times \{0,1\}\} \cup \{i\}$.

We can derive that the process $X = \partial_H(S \| K \| L \| R)$ satisfies the following recursive specification:

$$X = \sum_{d\in D} \lfloor r1(d) \rfloor^{\omega} \cdot X_d^1$$

$$X_d^1 = c2(d0) \cdot \left(i \cdot \sigma_d(X_d^1) + i \cdot c3(d0) \cdot s4(d) \cdot X_d^2 \right)$$

$$X_d^2 = c5(0) \cdot \left(i \cdot \sigma_d(X_d^3) + i \cdot c6(0) \cdot Y \right)$$

$$X_d^3 = c2(d0) \cdot \left(i \cdot \sigma_d(X_d^3) + i \cdot c3(d0) \cdot X_d^2 \right)$$

$$Y = \sum_{d\in D} \lfloor r1(d) \rfloor^{\omega} \cdot Y_d^1$$

$$Y_d^1 = c2(d1) \cdot \left(i \cdot \sigma_d(Y_d^1) + i \cdot c3(d1) \cdot s4(d) \cdot Y_d^2 \right)$$

$$Y_d^2 = c5(1) \cdot \left(i \cdot \sigma_d(Y_d^3) + i \cdot c6(1) \cdot X \right)$$

$$Y_d^3 = c2(d1) \cdot \left(i \cdot \sigma_d(Y_d^3) + i \cdot c3(d1) \cdot Y_d^2 \right).$$

To help the reader, we show the complete transition diagram in fig. 3.

Due to the fact that a complete cycle takes place in one time slice, the sender never receives a wrong bit, and so the specification could be simplified. We see that this protocol is more a PAR protocol (Positive Acknowledgement with Retransmission) than an ABP protocol.

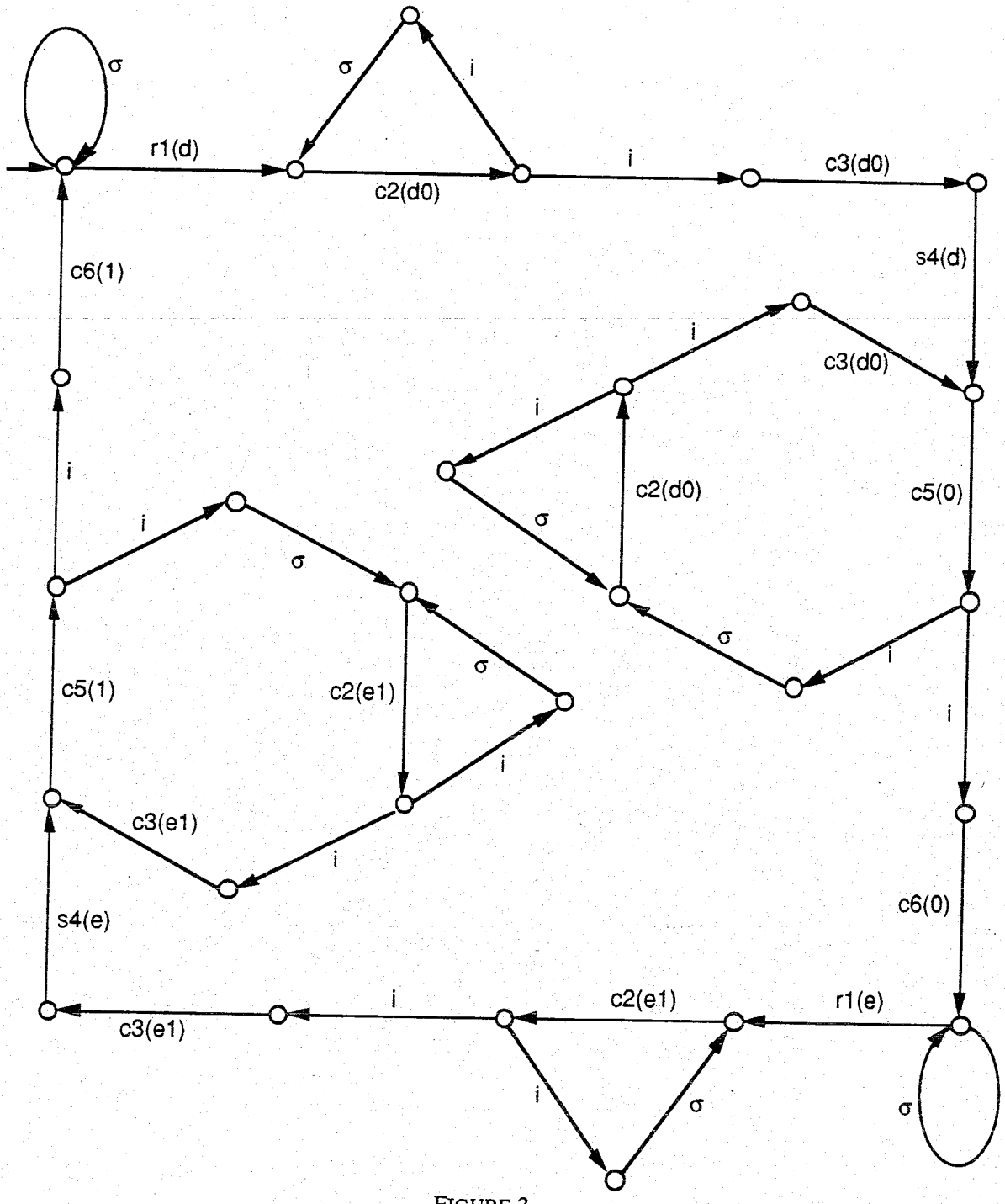


FIGURE 3.

After abstraction from internal actions, and using RSP, we obtain the following specification:

$$\tau_I(X) = \sum_{d \in D} \lfloor r1(d) \rfloor^{\omega} \cdot Z_d^1$$

$$Z_d^1 = \tau \cdot \sigma_d(Z_d^1) + \tau \cdot s4(d) \cdot Z_d^2$$

$$Z_d^2 = \tau\cdot\sigma_d(Z_d^3) + \tau\cdot\tau_I(X)$$

$$Z_d^3 = \tau\cdot\sigma_d(Z_d^3) + \tau\cdot Z_d^2.$$

Abstraction from time and using fair abstraction gives the desired result:

$$ABP = \tau_\sigma\circ\tau_I(X) = \tau\cdot\sum_{d\in D} r1(d)\cdot s4(d)\cdot ABP.$$

Notice: we need a generalisation of the fair abstraction rule in order to obtain this result.

7.3 ALTERNATING BIT PROTOCOL OF [BAB91a].

To conclude, we provide the discretisation of the Alternating Bit Protocol of [BAB91a] (page 186). We can again use fig. 2 in 7.2. As unit of time we take 0.0001 seconds.

Protocol:

$$ABP = \partial_H(S \parallel K \parallel L \parallel R)$$

where

sender

$$S = RM_0$$

$$RM_b = \sum_{v\in\mathbb{N}} \sum_{d\in D} r1(d)[v] \cdot SF_{db}$$

$$SF_{db} = s3(db)[10] \cdot RA_{db}$$

$$RA_{db} = \sum_{v\in\mathbb{N}} [\, r5(1\text{-}b)[v] + r5(e)[v]] \cdot SF_{db} + r5(b)[v]\cdot RM_{1\text{-}b}$$

channels

$$K = \sum_{v\in\mathbb{N}} \sum_{f\in D\times BOOL} r3(f)[v] \cdot K_f$$

$$K_f = \sum_{w\in(90,110]} (\, i[w]\cdot s4(e) + i[w]\cdot s4(f)) \cdot K$$

$$L = \sum_{v\in\mathbb{N}} \sum_{b\in BOOL} r6(b)[v]\cdot L_b$$

$$L_b = \sum_{w\in(90,110]} (\, i[w]\cdot s4(e) + i[w]\cdot s4(b)) \cdot L$$

receiver

$$R = RF_0$$

$$RF_b = \sum_{v\in\mathbb{N}} \sum_{d\in D} (r4(d(1\text{-}b))[v] + r4(e)[v]) \cdot SA_{1\text{-}b} + \sum_{d\in D} r4(db)[v] \cdot SM_{db}$$

$$SA_b = s6(b)[80] \cdot RF_{1\text{-}b}$$

$$SM_{db} = s2(d)[100] \cdot s6(b)[120] \cdot RF_{1\text{-}b}.$$

We can rewrite some equations as follows, using Nicollin-Sifakis functions:

$$RM_b = \sum_{d \in D} \lfloor r1(d) \rfloor^{\omega} \cdot SF_{db}$$

$$RA_{db} = \lfloor r5(1\text{-}b) + r5(e) \rfloor^{\omega} \cdot SF_{db} + \lfloor r5(b) \rfloor^{\omega} \cdot RM_{1\text{-}b}$$

$$K = \sum_{f \in D \times BOOL} \lfloor r3(f) \rfloor^{\omega} \cdot K_f$$

$$K_f = (90 \gg \lfloor i \cdot s4(e)) + i \cdot s4(f) \rfloor^{\omega} \gg 110) \cdot K$$

$$L = \sum_{b \in BOOL} \lfloor r6(b) \rfloor^{\omega} \cdot L_b$$

$$L_b = (90 \gg \lfloor i \cdot s4(e)) + i \cdot s4(b) \rfloor^{\omega} \gg 110) \cdot L$$

$$RF_b = \sum_{d \in D} \lfloor r4(d(1\text{-}b)) + r4(e) \rfloor^{\omega} \cdot SA_{1\text{-}b} + \sum_{d \in D} \lfloor r4(db) \rfloor^{\omega} \cdot SM_{db}.$$

8. CONCLUDING REMARKS.

We have proposed a theory on discrete time process algebra that extends ACP with the features of ATP in a setting consistent with that of ACPρ. Many options for further work remain. We mention some:

i. All extensions of ACP that have been developed can be integrated in the discrete time setting, such as state operators, process creation, signals, priorities, asynchronous communication (see [BAW90] for these ACP extensions.

ii. The proof theory of ACP_{dt} with initialisation abstraction requires further work. Interesting are principles dealing with infinite behaviour as RDP, RSP, AIP. In particular, the special case of regular processes needs attention because regularity requires a new definition in the presence of discrete time.

iii. The analysis of ACP_{dt} and its extensions as a term rewriting system may be worth attention.

iv. Like in the case of ACP, many more models than the bisimulation model are possible. Investigation of failures, ready, trace models as well as projective limit constructions will be worth while.

v. A theory that explains how discrete time process algebra can be used to develop simulation techniques for real time processes would be very interesting. We think that along this line the best perspective for application of discrete time process algebra can be found.

REFERENCES.

[BAB91a] J.C.M. BAETEN & J.A. BERGSTRA, *Real time process algebra*, Formal Aspects of Computing 3 (2), 1991, pp. 142-188.

[BAB91b] J.C.M. BAETEN & J.A. BERGSTRA, *A survey of axiom systems for process algebras*, report P9111, Programming Research Group, University of Amsterdam 1991.

[BAB91c] J.C.M. BAETEN & J.A. BERGSTRA, *Process algebra with signals and conditions*, report CS-R9103, CWI, Amsterdam 1991. To appear in Proc. NATO Summer School, Marktoberdorf 1990 (M. Broy, ed.), Springer Verlag.

[BAB92] J.C.M. BAETEN & J.A. BERGSTRA, *Real space process algebra*, report P9206, Programming Research Group, University of Amsterdam 1992.

[BAW90] J.C.M. BAETEN & W.P. WEIJLAND, *Process algebra,* Cambridge Tracts in Theor. Comp. Sci. 18, Cambridge University Press 1990.

[BEK84] J.A. BERGSTRA & J.W. KLOP, *Process algebra for synchronous communication,* Inf. & Control 60, 1984, pp. 109-137.

[BRHR84] S.D. BROOKES, C.A.R. HOARE & A.W. ROSCOE, *A theory of communicating sequential processes,* JACM 31 (3), 1984, pp. 560-599.

[GRO90a] J.F. GROOTE, *Specification and verification of real time systems in ACP,* in: Proc. 10th Symp. on Protocol Specification, Testing and Verification, Ottawa (L. Logrippo, R.L. Probert & H. Ural, eds.), North-Holland, Amsterdam 1990, pp. 261-274.

[GRO90b] J.F. GROOTE, *Transition system specifications with negative premises,* in: Proc. CONCUR'90, Amsterdam (J.C.M. Baeten & J.W. Klop, eds.), LNCS 458, Springer 1990, pp. 332-341.

[GM91] D.P. GRUSKA & A. MAGGIOLO-SCHETTINI, *A timed process description language based on CCS,* report TR-9/91, Universitá di Pisa 1991.

[HAN91] H. HANSSON, *Time and probabilities in formal design of distributed systems,* Ph.D. thesis, report DoCS 91/27, Uppsala University, 1991.

[HER90] M. HENNESSY & T. REGAN, *A temporal process algebra,* report 2/90, University of Sussex 1990.

[KLU91] A.S. KLUSENER, *Completeness in real time process algebra,* report CS-R9106, CWI Amsterdam 1991. Extended abstract in Proc. CONCUR'91, Amsterdam (J.C.M. Baeten & J.F. Groote, eds.), Springer LNCS 527, 1991, pp. 376-392.

[KLU92] A.S. KLUSENER, *Abstraction in real time process algebra,* to appear in Proc. REX Workshop on Real Time: Theory in Practice (J.W. de Bakker, C. Huizing, W.P. de Roever & G. Rozenberg, eds.), Mook 1991, Springer LNCS.

[LYV92] N. LYNCH & F.W. VAANDRAGER, *Action transducers and timed automata,* draft, March 1992.

[MIL89] R. MILNER, *Communication and concurrency,* Prentice-Hall 1989.

[MOT89] F. MOLLER & C. TOFTS, *A temporal calculus of communicating systems,* report LFCS-89-104, University of Edinburgh 1989.

[MOT90] F. MOLLER & C. TOFTS, *A temporal calculus of communicating systems,* in: Proc. CONCUR'90, Amsterdam (J.C.M. Baeten & J.W. Klop, eds.), LNCS 458, Springer 1990, pp. 401-415.

[NIS90] X. NICOLLIN & J. SIFAKIS, *The algebra of timed processes ATP: theory and application,* report RT-C26, IMAG, Grenoble 1990.

[NSVR90] X. NICOLLIN, J.-L. RICHIER, J. SIFAKIS & J. VOIRON, *ATP: an algebra for timed processes,* in Proc. IFIP TC2 Conf. on Progr. Concepts & Methods, Sea of Gallilee, Israel 1990.

[RER88] G.M. REED & A.W. ROSCOE, *A timed model for communicating sequential processes,* TCS 58, 1988, pp. 249-261.