A Relational Calculus for the Design of Distributed Algorithms

Een relationele calculus voor het ontwerpen van gedistribueerde algoritmen

(met een samenvatting in het Nederlands)

Proefschrift

ter verkrijging van de graad van doctor
aan de Universiteit Utrecht
op gezag van de Rector Magnificus, Prof. dr. J.A. van Ginkel
ingevolge het besluit van het College van Decanen
in het openbaar te verdedigen
op woensdag 27 september 1995 des middags te 12.45 uur

door

Frans Johan Rietman

geboren op 26 september 1967
te Kampen

Promotor: Prof. L.G.L.T. Meertens
Faculteit Wiskunde en Informatica

CIP-GEGEVENS KONINKLIJKE BIBLIOTHEEK, DEN HAAG

Rietman, Frans Johan

A relational calculus for the design of distributed
algorithms / Frans Johan Rietman. - Utrecht :
Universiteit Utrecht, Faculteit Wiskunde en Informatica
Proefschrift Universiteit Utrecht. - Met index, lit. opg.
- Met samenvatting in het Nederlands.
ISBN 90-393-0689-3
Trefw.: algoritmen

The research in this thesis was supported by the Netherlands Organisation for Scientific Research (NWO) under grant 025-62-518, NFI-project "Specification and Transformation of Programs" (STOP).

Contents

Acknowledgements

Writing a thesis is the final state of a period of four years of intensive research, mostly in cooperation with other people. I would like to thank them for all the support.

First of all, I want to thank Lambert Meertens for offering me the OIO position at the Department of Computer Science, Utrecht University, and for supervising me for the last four years. I pity him for his efforts to convince me of my (potential) faults; and still, I'm not convinced in all cases. There is no doubt, however, that he was often right, in particular when he accused me of being stubborn.
There were several meetings which contributed to my research. Among them are the STOP workshops and Summer schools on Ameland, and the weekly (well, almost) meetings at Utrecht (Constructive Algorithmics Club, Woensdag Middag Club) and Eindhoven (Mathematics of Program Construction Group). During those meetings I was introduced to the fields of Transformational Programming, Relational Programming and Distributed Programming. Furthermore, the monthly meetings of Mathematics of Programming (MoP) are appreciated. They provided a forum which enabled me to present my own work, and to get familiar with other approaches to doing research.

Furthermore, I would like to mention a few people for their specific contributions to this thesis[1]:
All the people from 'de eerste'[2] for creating such a nice working environment, and, more importantly, for having time to go to the movies, for playing skwosj and for the numerous things I can't remember (sometimes, it was quite late).
Carroll Morgan for initiating the research on chronicles together with Lambert, and for the fruitful and pleasant time during his visit to Utrecht in 94/95.
Chritiene Aarts and Henk Doornbos for helping me when I had serious problems with the relational algebra, for having stimulating discussions, and just for being good friends.
For nearly four years, Rob Udink was my roommate. I don't want to give a complete summary of all the interesting, deep, personal, etc. talks we had, but simply characterise those conversations as most valuable for me during the time I spent in Utrecht. Thanks.
Joost Kok for guiding me through the theory on semantics and dataflow networks.

[1] "... and in no [redacted] way is this in order of priority" – *Guns n' Roses*, cover *Use your illusion.*
[2] This name determines a group, not a place.

Maarten Pennings for providing a lot of useful LaTeX macros and for helping me to understand and circumvent the typical error messages and warnings produced by this 'thesis preparation system'. I am hardly able to understand something like:

```
\def\mynewtheorem#1{\newtheorem{mcp@#1}[equation]{#1}%
  \newenvironment{#1}[1]%
    {\begin{mcp@#1}\mbox{\it ##1}\rm\par}%
    {\ifvmode\mbox{}\else\\\fi$\Box$\end{mcp@#1}}}
```

I was not—and probably never will be—able to invent this myself.
Maria Ferreira is thanked for explaining the theory of well-ordered sets.
The members of the reading committee, Prof. Dr R.C. Backhouse, Prof. Dr W.H. Hesselink, Prof. Dr J.N. Kok, Dr C.C. Morgan and Prof. Dr S.D. Swierstra are acknowledged for reviewing this thesis.

The object on the cover is generated by the command:

```
xmartin -f cp1,3,1,4 -zoom 230 -a 18.3041 -b -6.10138 -x 20 -y 30 -coord ar -move ne,100
```

For those who are puzzled: the picture doesn't mean anything. It decorated the background of my terminal during the final year at the department.

Chapter 1

Introduction

1.1 Historical background and related work

Networks of processes can be modelled as a dataflow network. Such networks are represented by a graph consisting of nodes and directed arcs. Each node corresponds to a process, while each arc corresponds to a (buffered) channel. Kahn's insight in his pioneering paper [Kah74] was that for deterministic processes the operational behaviour of such networks could be captured denotationally in a very simple and elegant way, using elementary domain theory. He modelled the behaviour of a process by a continuous history function. The operational behaviour of the whole network could be obtained by setting up a system of equations (one for each channel in the network) and solving the system by taking the least fixpoint. This result is called the *Kahn Principle*.
Since then, a number of people have tried to extend Kahn's work to networks with non-deterministic processes: instead of modelling processes by history functions, processes were modelled by history *relations*. Unfortunately, the construction of feedback loops introduces anomalies, as was shown by Keller [Kel78] and a few years later by Brock & Ackerman [BA81]. Additional properties involving timing or causality aspects are needed to circumvent these problems.

Based on work by Misra [Mis90], Abramsky [Abr90] generalised the Kahn Principle to a large class of non-deterministic systems. In a very general setting (it includes the usual models based on linear traces), he describes processes in a network as *sets* of functions. Each such function obeys the causality condition that an additional output event can only be the result of past input events. Then, it is shown that the (non-deterministic) operational behaviour of a network is described denotationally by the *set* of fixpoints, each of which is specified by ordinary Kahn semantics.
Despite the carefully designed semantics of non-deterministic networks, Abramsky's work does not result in a calculus. Stark [Sta90] also states and proves a generalised Kahn principle; he even relates his own result to that of Abramsky [Abr90]. In a subsequent paper, Stark [Sta92], a set of *equational laws* (about, for example, buffering and feedback) is given, together with a completeness result stating that those equational laws are complete and

sound with respect to a dataflow calculus of built-in processes, process-forming operations, transition axioms and inference rules. Despite the elegance of the obtained calculus, we could not find an application in the literature.
Another good example of a carefully designed calculus is the work of Broy [Bro90]. He concentrates on the notions of specification and several forms of refinement. The behaviour of deterministic processes is represented by a *stream processing function* which is continuous and prefix monotonic. Prefix monotonicity is the causality property that more input can only produce more output; no previous output is retracted. Non-deterministic processes (specifications) are described by predicates that characterise a set of functions representing their set of possible behaviours. In addition to these processes, several combining forms such as sequential composition, parallel composition and feedback are given. Broy argues that all calculations on refinement can be performed completely using functions.
Other calculi, which have reached the stage of being widely applied, are UNITY [CM89] (with operational semantics similar to action systems, Back [Bac93]) and CSP [Hoa85] (and other process algebras, Milner [Mil89] or Baeten & Weijland [BW90]). These calculi incorporate non-determinism. The methodology of UNITY is adapted to the development of programs for a variety of architectures and applications. The foundation on which program development is based is a theory similar to temporal logic. Programs and specifications are in different formalisms. Only specifications can be refined, after which a program is obtained. A refinement proof often boils down to reasoning about execution sequences and program states. A UNITY program is not a network of processes. To develop a distributed program, a UNITY programmer has to partition his program into isolated processes. The theory of CSP, based on a model of traces, is better suited for distributed programming. One of the reasons is that the main objects of manipulation are (channel-oriented) processes rather than (state-oriented) statements. Another advantage over UNITY is that specifications and programs in CSP are in the same formalism; no mapping from a refinement to a program is needed.

1.2 Transformational programming

The approach to the derivation of distributed algorithms we are heading for is *transformational programming*. Specifications, described in a formal system, are viewed as correct algorithms which can be highly inefficient or contain non-implementable parts. By successive meaning-preserving (or meaning-refining) transformation steps, formulated in *transformation rules*, those specifications are transformed into correct and efficient algorithms built from implementable primitives. Therefore, the process of applying transformation rules can stop if all non-implementable parts are removed. Since each transformation step preserves (or refines) the meaning of the algorithm, the overall result of a sequence of transformation steps is correct again. Because the concept of efficiency depends on the actual implementation in some language on some architecture, and we do not want to commit ourselves unduly, efficiency considerations have to be informal.

To be able to state and apply transformation rules, a transformational programming calculus should possess the following two properties. First, to perform calculations, it should be possible to express both specifications and algorithms in the language used by the calculus; second, to perform understandable calculations, the notation used for algorithms should be concise and the rules should be simple but powerful.
Transformational programming was advocated by Backus [Bac78] in his ACM Turing Award Lecture. He proposed an algebra of programs and combining forms:

> *This algebra can be used to transform programs and to solve equations whose "unknowns" are programs in much the same way one transforms equations in high school algebra. These transformations are given by algebraic laws and are carried out in the same language in which programs are written. Combining forms are chosen not only for their programming power but also for the power of their associated algebraic laws.*

Among the algebraic laws mentioned by Backus are rules such as distributivity, associativity and idempotency, and monotonicity with respect to some (refinement) order.
Good recent examples of transformational programming for program derivation and the development of new theory can be found in the theses of Fokkinga [Fok92], Jeuring [Jeu93] and Meijer [Mei92].

1.3 Goals and contents of this thesis

In this thesis we look at the problem of distributed, communicating processes. There are several varieties of distributed algorithms, such as synchronous versus asynchronous algorithms. We strive for a mathematical model that can serve as an abstraction for all these varieties. The difference should correspond to different constraints imposed on a general unifying mathematical framework, leading to different rules in the calculus.
One goal is the development of a calculus that is suited for dealing with non-determinism in a natural way. The final calculus should provide a box of powerful tools, containing useful primitive processes and concise transformation rules.
In the spirit of Backus, the second goal is to avoid the explicit use of states; states are inherited from the von Neumann programming style. During calculations, especially in a calculus for distributed algorithms, the presence of a decentralised state clouds the sky.
A third goal is the abstraction from time: only the ordering of events is important, not the exact timing. As announced, the difference between several assumptions on time (for example, discrete versus continuous) only occurs in the difference between sets of rules in the calculus. Therefore, no single rule in the calculus will ever refer to explicit time.
We do not intend to be exhaustive in presenting a new calculus. Investigations are still in progress, and the set of transformation rules and primitive processes is subject to change.

To introduce non-determinism in a calculus for the design of distributed algorithms, we start with a well-established relational algebra, and extend this algebra with axioms and

definitions for constructs needed in a calculus for the design of distributed algorithms. Calculations in the model justify the axioms introducing the new constructs.
In the model we propose, processes are modelled as relations that relate timed input streams to timed output streams; those timed streams will be called *chronicles*. This contrasts with most related work in which *events* on channels are merged to form a *trace* that describes a behaviour of the system. The hope is that, by using relations as the basis, a simple calculus with a high level of abstraction will result.
All processes calculate simultaneously throughout time, and communicate with other processes in a synchronous way. Buffering of messages is not assumed: a buffer is just an ordinary process in some network of processes, and has to be mentioned explicitly, built on top of the synchronous system. Consequently, if a process is not ready to accept a message, other processes can not send to them and the communication 'blocks'. One has to use explicit protocol schemes (buffered processes) to avoid this blocking, resulting in asynchronous communication.
Processes can be connected in networks of processes using composition constructions. These compositions represent dedicated channels connecting (pairs of) processes and describe the topology of the network. The connection therefore represents the flow of data, which contrasts flow of control.

The thesis consists of five parts. We aim at a small application in the final part. All preceding parts introduce useful concepts for the design of distributed algorithms, exploiting a calculus of relations.
In Part I, we summarise the relational algebra and specify the particular model we are going to use in our calculations. Chapter 2 explains our view of processes and gives examples of ways to compose processes to obtain larger networks. Processes will be mathematically described as binary (input-output) relations, and relational composition constructs can be used to describe the topological structure of networks. Then, Chapter 3 discusses the raw relational calculus. It is introduced by giving a set of axioms. This axiomatisation is sound, witnessed by a set-theoretical model. To get a more realistic calculus with respect to implementation, also a demonic composition operator is discussed. It is shown that the angelic composition from the raw relational calculus can easily be transformed into its demonic counterpart. Apart from the feedback construct and two preservation problems related to feedback, Chapter 3 contains no topics concerning distributed programming. Finally, Chapter 4 deals with the particular model for the relational algebra: the model of chronicles. In this chapter, important concepts such as message domain and time domain are defined.
Part II introduces two important functions for performing actions on messages and on time. In Chapter 5, the function postcompose is defined and investigated. It prescribes actions on the messages and is used to define some simple processes. Actions on the time domain can be specified by the function precompose, Chapter 6. This gives us the possibility to reason about shifts in time, or to restrict the time domain to some part we want to observe. Despite the fact that the functions postcompose and precompose are dual, we will only axiomatise the function postcompose in the relational calculus. The reason to neglect the

function precompose is that in the final calculus we want to avoid explicit time or actions on time.
In Part I we encounter several anomalies with respect to feedback. Our solution to these anomalies is presented in Part III: the property of causality. First, Chapter 7 records the consequences and properties of causal processes, in particular when these processes are placed in a feedback loop. After having motivated the importance of causality, preservation rules for the composition constructions are derived in Chapter 8. Chapter 9 strengthens some of the preservation rules by weakening their conditions.
Some useful primitive processes in the calculus are identified in Part IV. Three processes for delaying the input or taking some prefix are defined in Chapter 10. These processes pave the way for the definition of a synchronisation process in Chapter 11. This process is investigated in detail, resulting in new axioms imposed on the processes of Chapter 10. The final chapter of Part IV, Chapter 12, defines and explores the buffer process, and mentions some extensions and related processes.
The buffer and its extensions of Chapter 12 are the key processes in Part V, which brings all results obtained in the previous chapters together in an effort to derive a well-known communication protocol by calculation. Before doing so, a small instantiation of the general calculus is made in Chapter 13 in order to be able to reason about typed procs. Still, many of the calculations in Chapter 14, where we derive the communication protocol, could be done without the burden of type checking.

1.4 On nomenclature

Before starting, some remarks are made on the nomenclature of assumptions, definitions and results. Because we want to abstract from the model, and in particular from the time aspects, we will make a clear distinction between assumptions made in the *model*, and assumptions made in the *calculus*; and similar for definitions, theorems, etc.
For the model, where time plays an important role, the following terminology is used. For assumptions in the model: **Assumption**; for definitions in the model concerning time aspects: **Characterisation**; for all other definitions in the model: **Definition**; for results depending on time aspects and which are *not* meant to be axiomatised: **Proposition**; for results depending on time aspects and which *are* meant to be axiomatised: **Property**; and, finally, for all other results: **Theorem**. At the end of Parts II, III and IV, the obtained properties are axiomatised.
In the calculus, common nomenclature occurs: **Axiom**, **Definition**, **Lemma**, **Theorem** and **Corollary**. They all speak for themselves. In the proof of a **Lemma** or a **Theorem**, the use of statements from the model level is not allowed. An exception is the use of **Properties** at places where they have not yet been axiomatised.
Not all the proofs of all propositions, properties, lemmas and theorems are given: several proofs bearing great similarity to other proofs have been omitted.

Part I

Basics

Chapter 2

Processes and networks

This chapter introduces our view of processes and networks. We motivate why processes are regarded as binary relations. Furthermore, the need for several composition constructions is explained. Part of the discussion below also appeared in Rietman [Rie93b].

2.1 Processes

A possible model of a distributed program is that of a collection of communicating processes connected by channels, for example as in Figure 2.1:

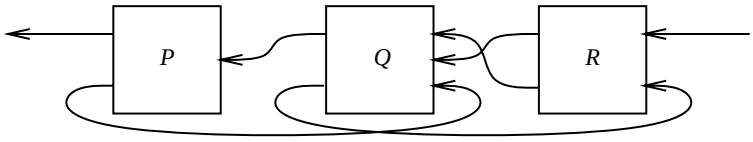

Figure 2.1: Small network

We will explain how we embed networks like these in an algebra of binary relations. Let $\mathcal{A}$ be the set of processes. Each process has input ports and output ports. A channel connects an output port to an input port. Processes communicating with the 'outside world' can be thought of as having channels connected to 'outside' ports.
The process P in Figure 2.1 corresponds to a relation that relates an input channel, which happens to be a singleton, to an output channel, which is a pair. In the mathematical model, we abstract from any notion of physical lay-out: only the topology of the network—which ports are connected to which ports—is important. We assume that the left-hand side ports are output ports, and the right-hand side ports are input ports.
A channel carries a stream of values. These streams are called *chronicles* and are defined in Chapter 4. In short, they are total functions of type $\mathcal{C} := \mathcal{M} \leftarrow \mathcal{T}$, where $\mathcal{T}$ is some time domain and $\mathcal{M}$ is some 'well-typed' message domain. Identifiers f, g and h range over $\mathcal{C}$.

If the possible input chronicles are in the set $A \subseteq \mathcal{C}$, and the possible output chronicles are in the set $B \subseteq \mathcal{C}$ we say that P has type B from A. Observe that typing is *not* unique. For example, every process has type $\mathcal{C}$ from $\mathcal{C}$.

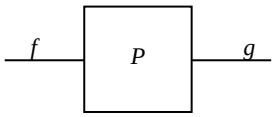

Figure 2.2: Process P relates chronicle $f \in B$ to chronicle $g \in A$

Mathematically, Figure 2.2 is expressed in the model by $f \langle P \rangle g$. Several different chronicles f can be related by relation P to g, or, symmetrically, several different chronicles g can be related to f. Typing of P is denoted by $P \in B \sim A$.
Allowing relations in the model introduces *non-determinism*. There are various reasons for allowing non-determinism in the model; one reason is the possible delay of a process. For example: given two input values to an adder at some moment, it is completely determined what the result of the addition will be. However, it might be unknown when this result occurs on the output channel, due to an unknown delay in the adder. In this sense, the adder is non-deterministic. Another reason for non-determinism is the subject of *under-specification*.
Specifications and programs are in the same formalism: a process can be a specification as well as a program. Programs are, in fact, specifications that are deemed to be (directly) 'implementable'. We are not concerned here with a precise definition of implementability, which may depend on the specifics of some programming language. We need ways, though, to combine programs into new programs, that is, ways to compose relations in such a way that implementability is preserved. These composition methods, which can be interpreted as ways to obtain connection patterns of some network, are the subject of the following sections; in Chapter 3 the compositions will be formally introduced. Often, properties of the new processes can be derived from properties of the constituents by means of preservation properties of the composition operator.

2.2 Sequential composition

To connect two processes sequentially the relational angelic composition is used. In this way, processes of arbitrary types can be combined. The calculus provides tools to identify which types are meaningful. This composition looks like *chaining* in CSP [Hoa85].

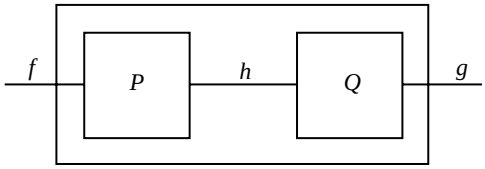

Figure 2.3: Sequential composition of P and Q

The process $P \circ Q$ relates some output chronicle f to some input chronicle g if and only if there exists a chronicle h such that P relates f to h and Q relates h to g, Figure 2.3. The identity of sequential composition I can be thought of as a bundle of (arbitrarily many) wires. The direction (from right to left) will be fixed by the notion of *functionality* in Chapter 3. In that chapter, we will also introduce a relational *demonic* composition, which is actually the composition used in implementations.

2.3 Parallel composition

To combine two processes that compute in parallel, that is, the two processes do not have (direct) interaction with each other, parallel composition is used. The parallel composition of processes P and Q is denoted by $P \parallel Q$. The process $P \parallel Q$ relates an input pair of chronicles to an output pair, Figure 2.4.

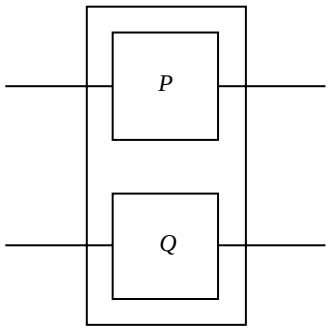

Figure 2.4: Parallel composition of P and Q

A warning should be given here. The parallel composition used in this calculus is not similar to merge operators used in CSP or in dataflow networks. In those theories the parallel operator expresses (synchronised) communication of the operands. In our theory, parallel composition is similar to product in category theory and expresses simultaneous computation of the operands without (direct) communication. In fact, all composition constructions presented in the theory express simultaneous computation. The composition denotes the way these processes are *connected*.

2.4 Split and projections

Elementary actions on pairs of channels can be performed by split and the projections. Split combines two processes P and Q in parallel, attached to the same input channel: $P \triangle Q$. The projections $\ll$ and $\gg$ select one of the input channels. The left projection and the split of P and Q are depicted in Figure 2.5.

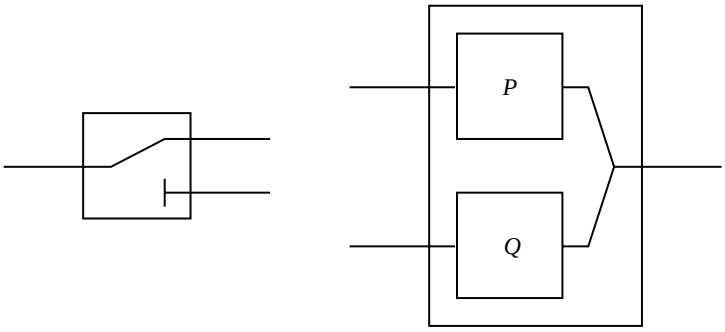

Figure 2.5: Left projection $\ll$, and split $P \triangle Q$

At this point there is a clear divergency between the pictures and the theory: when the processes in Figure 2.5 are composed sequentially, one is tempted to conclude that the result is the process P. But this is, in general, not the case. The freedom to write $P \triangle Q$ for all P and Q (without the need for type checking) forces us to take into account the type of Q when trying to reduce the combination of the processes in Figure 2.5. These type considerations are exactly what we have in mind: no type checking when writing some expression. The type of (part of) the expression emerges from the calculation.

2.5 Feedback

One of the most important components of a relational calculus for the design of distributed algorithms is the feedback operator. Its importance is comparable to the importance of the concept of loops in iterative programming and recursion in functional programming. However, *the* feedback operator does not exist. Therefore, we want to have a general one in the sense that it should be possible to construct all other possible feedback constructions from it. This motivates the choice to feed back *all* the output channels. Then, at the input side, every possible combination of the fed-back output channels can be chosen to contribute to the input. The feedback of process P is denoted by P^{σ}.

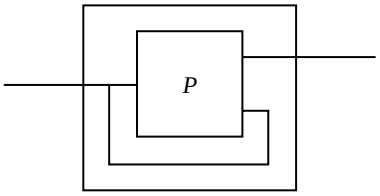

Figure 2.6: Feedback circuit P^σ of P

The rich structure of the algebra introduced in Chapter 3 will enable us to *define* the feedback operator. No new axioms are assumed to capture the notion. Many of the desired properties of feedback, such as the computation rules, follows from its definition. It would be naive, though, to think that all the problems other researchers in the field encountered are circumvented.

Chapter 3

The relational algebra

This chapter summarises the algebraic framework in which the calculations are made. This framework can also be found in Backhouse *et al.* [BdBH+91], which is based on the pioneering work by Tarski [Tar41], and related to work by Maddux [Mad91] and Schmidt & Ströhlein [SS93]. Compared to [BdBH+91] there are only some minor differences in notations and terminology. The calculations that are shown here merely serve as illustrations.
In Section 3.1, the relational algebra is presented. As the basic sequential composition we will encounter angelic relational composition. However, in implementations the composition tends to be demonic. When comparing angelic composition with demonic composition, it turns out that the former 'delivers a result' whenever a sensible result exists. In contrast, the result of demonic composition is undefined whenever the possibility of failure exists. However, angelic composition cannot be implemented effectively. Then why use angelic composition? The reason is that the calculational properties of demonic composition are not as nice as those of angelic composition. Some of the disadvantages are the non-distribution of demonic composition over disjunction and conjunction (from the left), due to its anti-monotonic behaviour. Moreover, there are typing problems in using demonic composition. Angelic composition, on the other hand, has nice distribution rules with respect to disjunction, and reasonable conditions for the distribution over conjunction. Furthermore, there is no need for typing considerations when composing two processes with angelic composition. Last but not least, angelic composition is easily transformed into demonic composition by requiring totality of the processes involved.
After introducing the framework, useful concepts, such as functionality and totality, are defined in Section 3.2. The concept of functionality (or determinism) is an important one in our calculus: functional processes obey important distribution laws not satisfied by processes in general. Totality is used to transform the angelic composition of the calculationally derived processes into the demonic composition of an implementation. Section 3.3 will be devoted to this matter. In Section 3.4 the Principle of Extensionality is explained and axiomatised. Finally, Section 3.5 discusses two preservation problems of feedback.

3.1 The algebraic framework

The framework is a relational algebra $(\mathcal{A}, \sqsubseteq, \circ, I, {}^{\cup}, \ll, \gg)$. It is introduced in four layers, connected by interfaces.
A model for the axiomatisation is the structure of the set-theoretical relations over some universe. That is, the objects of $\mathcal{A}$ are sets of pairs of elements from the universe. Together with the introduction of new objects in $\mathcal{A}$ and new operators for constructing new objects from old ones, interpretations (or justifications) in the model are given. In each interpretation the quantification is over all f, g, h and i from the universe. To express that a pair (f, g) is an element of relation R the notation $f \langle R \rangle g$ is used. In operational terms this says: for input g (describing a complete input history) a possible output of R is f (describing a complete output history). In Chapter 4 the elements of the universe are specified in more detail.

3.1.1 Lattice

The first layer is the structure of a lattice, see for example Davey & Priestly [DP90]. Let $\mathcal{A}$ be a set, the elements of which are to be called *procs* (from processes). The identifiers P, Q, etc. range over $\mathcal{A}$. On $\mathcal{A}$ the structure of a complete, universally distributive, complemented lattice is imposed:

Axiom 3.1 *Lattice*

$$(\mathcal{A}, \sqsubseteq)$$

is a complete, universally distributive, complemented lattice. Here, 'lattice' means that the join operator, denoted by $\sqcup$ and called *cup*, and the meet operator, denoted by $\sqcap$ and called *cap*, are associative, commutative and idempotent, binary infix operators with unit elements $\perp\!\!\!\perp$ and $\top\!\!\!\top$, respectively.
'Complete' means that the extrema $\sqcup(\, P \;:\; P \in \mathcal{B} \;:\; P\,)$ and $\sqcap(\, P \;:\; P \in \mathcal{B} \;:\; P\,)$ defined in the usual way exist for all bags $\mathcal{B}$ of procs.
'Universally distributive' means: for all bags $\mathcal{B}$ of procs:

$$P \sqcap \sqcup(\, Q \;:\; Q \in \mathcal{B} \;:\; Q\,) \;=\; \sqcup(\, Q \;:\; Q \in \mathcal{B} \;:\; P \sqcap Q\,)$$

$$P \sqcup \sqcap(\, Q \;:\; Q \in \mathcal{B} \;:\; Q\,) \;=\; \sqcap(\, Q \;:\; Q \in \mathcal{B} \;:\; P \sqcup Q\,)$$

'Complemented' means that the rules of Contradiction and Excluded Middle are valid:

$$P \sqcap \neg P = \perp\!\!\!\perp$$

$$P \sqcup \neg P = \top\!\!\!\top$$

□

The interpretations in the set-theoretical model are:

$$f\,\langle P \sqcup Q\rangle\,g \;\equiv\; f\,\langle P\rangle\,g \;\vee\; f\,\langle Q\rangle\,g$$

$$f\,\langle \bot\!\!\bot\rangle\,g \equiv \mathit{False}$$

$$f\,\langle P \sqcap Q\rangle\,g \;\equiv\; f\,\langle P\rangle\,g \;\wedge\; f\,\langle Q\rangle\,g$$

$$f\,\langle \top\!\!\top\rangle\,g \equiv \mathit{True}$$

$$P \sqsubseteq Q \;\equiv\; \forall(\, f, g \;:\; f\,\langle P\rangle\,g \;:\; f\,\langle Q\rangle\,g\,)$$

$$f\,\langle \neg P\rangle\,g \equiv \neg(f\,\langle P\rangle\,g)$$

The order $\sqsupseteq$, meaning $P \sqsupseteq Q \;\equiv\; Q \sqsubseteq P$, will also be used. The distributivity properties in Axiom 3.1 are also referred to as: $P\sqcap$ is (universally) *cupjunctive* and $P\sqcup$ is (universally) *capjunctive*.

The complemented lattice structure is well known from the literature. For example, the predicate calculus, Dijkstra & Scholten [DS90], embeds the structure. We call it a *plat*, standing for power set lattice. Calculations in the structure are referred to by the hint 'plat calculus'.

Before continuing, some remarks are made on operator precedence. First, the connectives in the predicate calculus ($\equiv$, $\Leftarrow$, $\Rightarrow$, $\vee$ and $\wedge$) have lower precedence than all the operators in the relational algebra. Next, the operators in the lattice structure $=$, $\sqsubseteq$ and $\sqsupseteq$ all have equal precedence. Also $\sqcup$ and $\sqcap$ have equal precedence, but higher than the other binary operators in the lattice structure. Because $\neg$ is a unary operator, it has the highest precedence of all the operators in the lattice. So, $\neg P \sqcup Q \sqsubseteq R$ should be parsed as $((\neg P) \sqcup Q) \sqsubseteq R$.

In a complete lattice, every monotonic function F has a least fixpoint; the least fixpoint of F is denoted by μF. We record the characterising property of the least fixpoint μF. First, the computation rule:

$$F.\mu F = \mu F$$

and second, the induction rule:

$$\forall(\, P \;::\; \mu F \sqsubseteq P \Leftarrow F.P \sqsubseteq P\,)$$

We find the characterisation above inadequate for our calculations: in several applications of the induction rule the antecedent turns out to be too strong. Therefore, we rewrite the characterisation of μF:

Theorem 3.2 *Fixpoint Characterisation*
The least fixpoint μF of the monotonic function F is characterised by:

$$\forall(\, P \;::\; \mu F \sqsubseteq P \;\equiv\; F.(P \sqcap \mu F) \sqsubseteq P\,)$$

□

The implication $\Rightarrow$ equivales $F.\mu F \sqsubseteq \mu F$; the implication $\Leftarrow$ then equivales the induction rule. These two conjuncts express that μF is the least solution of the inequation $X :: F.X \sqsubseteq X$. Then, we exploit the Knaster-Tarski Fixpoint Theorem [Tar55][1], which states that the equation $X :: F.X = X$ and the inequation $X :: F.X \sqsubseteq X$ both have the same least solution for monotonic F.
The least fixpoint operator μ is defined in any complete lattice. In the rest of this thesis, several procs, relations or ordinary sets are defined using the schema of Theorem 3.2.

3.1.2 Sequential composition

The second layer is the monoid structure for sequential composition:

Axiom 3.3 *Sequential composition*

$$(\mathcal{A},\ \circ,\ I)$$

is a monoid, that is, $\circ$ is an associative binary infix operator with unit element I. The interface with the plat structure is: $\circ$ is coordinate-wise universally cupjunctive. That is, for bags $\mathcal{B}$ and $\mathcal{C}$ of procs,

$$(\sqcup\mathcal{B}) \circ (\sqcup\mathcal{C}) = \sqcup(\ P, Q\ :\ P \in \mathcal{B} \wedge Q \in \mathcal{C}\ :\ P \circ Q\ )$$

□

In the model, sequential composition is the *angelic* relational composition:

$$f\,\langle P \circ Q\rangle\,g\ \equiv\ \exists(\ h\ ::\ f\,\langle P\rangle\,h \wedge h\,\langle Q\rangle\,g\ )$$

$$f\,\langle I\rangle\,g\ \equiv\ f = g$$

Sequential composition has higher precedence than the binary operators in the lattice structure. So, $P \sqcap \neg Q \circ R$ is to be read as $P \sqcap ((\neg Q) \circ R)$.
From Axiom 3.3, it follows that $\circ$ is monotonic with respect to $\sqsubseteq$ and has $\perp\!\!\!\perp$ as a left and right zero. Another consequence of the interface of sequential composition with the lattice is the existence of *factors*:

Definition 3.4 *Factors*
The *right factor* $P \backslash R$ is defined by:
a. $\quad P \circ Q \sqsubseteq R\ \equiv\ Q \sqsubseteq P \backslash R$
And the *left factor* R/Q is defined by:
b. $\quad P \circ Q \sqsubseteq R\ \equiv\ P \sqsubseteq R/Q$
□

[1] This theorem is a lattice-theoretical generalisation of the set-theoretical theorem in Knaster [Kna28].

In the model, factors correspond to a universal quantification. Their interpretation can be derived using ordinary predicate calculus from the interpretation of sequential composition:

$$f\,\langle P\backslash Q\rangle\,g \;\equiv\; \forall(\,h\;:\;h\,\langle P\rangle\,f\;:\;h\,\langle Q\rangle\,g\,)$$

$$f\,\langle P/Q\rangle\,g \;\equiv\; \forall(\,h\;:\;g\,\langle Q\rangle\,h\;:\;f\,\langle P\rangle\,h\,)$$

Factors have a higher priority than sequential composition. Definitions 3.4a and 3.4b are instances of Galois connections, see for example Schmidt [Sch53] and more recently Aarts [Aar92]. The main corollary of Definition 3.4 is formed by the so-called *cancellation rules*:

Theorem 3.5 *Cancellation*

a. $P \circ P\backslash Q \sqsubseteq Q$

b. $P/Q \circ Q \sqsubseteq P$

□

To give an illustration of a calculation in the relational algebra and to demonstrate our proof style, the result $\top\!\!\top \circ \top\!\!\top = \top\!\!\top$ is proved:

Theorem 3.6 $\top\!\!\top$ *is idempotent*

$$\top\!\!\top \circ \top\!\!\top = \top\!\!\top$$

Proof by mutual inclusion:

$$\begin{array}{ll} & \top\!\!\top \\ \sqsupseteq & \quad \{\ \top\!\!\top \text{ is top element of the lattice}\ \} \\ & \top\!\!\top \circ \top\!\!\top \\ \sqsupseteq & \quad \{\ \top\!\!\top \text{ is top element: } \top\!\!\top \sqsupseteq I\text{; monotonicity of composition}\ \} \\ & I \circ \top\!\!\top \\ = & \quad \{\ I \text{ is identity of composition}\ \} \\ & \top\!\!\top \end{array}$$

□

Although the relational calculus by itself has no left-right bias, in our use of it, the input-output direction is typically assumed to be from right to left. Thus, our sequential composition is 'backwards'. However, it is only by defining what is meant by *functionality*, Subsection 3.2.2, that this direction is fixed.
The transitive and reflexive closure P^* of a proc P is defined by:

Definition 3.7 *Transitive and reflexive closure*
Let P^n, for $n \in \mathbf{N}$, be defined by:

a. $P^0 \triangleq I$

b. $P^{n+1} \triangleq P \circ P^n$

The transitive and reflexive closure of P is defined as:

c. $P^* \triangleq \sqcup(\, n \;:\; n \in \mathbf{N} \;:\; P^n \,)$

□

There are several equivalent definitions for the transitive and reflexive closure of proc P. It could be defined as the least solution of the fixpoint equation $X \;::\; X = I \sqcup P \circ X$, see Kleene [Kle52] and Rogers [Rog67]. The definition given above is probably the most familiar one, and suitable for our purposes.

3.1.3 Reverse

The third layer is the reverse structure. It is axiomatised as follows:

Axiom 3.8 *Reverse*

$$(\mathcal{A}, {}^{\cup})$$

introduces the unary postfix operator $^{\cup}$. The interface with the lattice structure is the following Galois connection:

$$P^{\cup} \sqsubseteq Q \;\equiv\; P \sqsubseteq Q^{\cup}$$

The interface with the monoid structure of sequential composition is:

$$(P \circ Q)^{\cup} = Q^{\cup} \circ P^{\cup}$$

□

In the model, reverse switches the input ports and the output ports. It is a generalisation of the inverse of functions:

$$f \,\langle P^{\cup} \rangle\, g \equiv g \,\langle P \rangle\, f$$

Because reverse is an unary operator, it has the same precedence as negation. Therefore, $P \sqcap Q \circ R^{\cup}$ is to be read as $P \sqcap (Q \circ (R^{\cup}))$.
From the interface with the lattice structure it follows that $^{\cup}$ is its own inverse. Moreover, reverse is universally cupjunctive and capjunctive. This implies $\bot\!\!\bot^{\cup} = \bot\!\!\bot$, $\top\!\!\top^{\cup} = \top\!\!\top$, and the monotonicity of reverse. The expression $\neg P^{\cup}$ does not need to be parenthesised because it also follows that reverse commutes with negation. From the interface with the monoid structure of sequential composition it follows that $I^{\cup} = I$.

3.1.4 Parallel composition

The fourth layer is the layer of parallel composition. The introduction of parallel composition differs from the axiomatisation of sequential composition. First, we assume the existence of the projections $\ll$ and $\gg$. Second, parallel composition and split are defined in terms of those projections and other elements of the calculus such as $\sqcap$ and sequential composition. Finally, the axiomatisation of the layer is completed.

Axiom 3.9 *Parallel composition*

$$(\mathcal{A},\ll,\gg)$$

postulates the existence of two procs $\ll$ and $\gg$. The proc $\ll$ is called 'left projection'; the proc $\gg$ is called 'right projection'. Before continuing with Axiom 3.9, two binary operators on procs, $\triangle$ and $\|$, referred to as 'split' and 'parallel', are defined:

Definition 3.10 *Split and parallel composition*

a. $$P \triangle Q \ \triangleq\ \ll^{\cup} \circ P \sqcap \gg^{\cup} \circ Q$$

b. $$P \,\|\, Q \ \triangleq\ (P \circ \ll) \triangle (Q \circ \gg)$$

(End of Definition 3.10)
□

The axiomatisation of the layer is completed:

c. $$I \sqsupseteq I \,\|\, I$$

d. $$(P \triangle Q)^{\cup} \circ R \triangle S \ =\ P^{\cup} \circ R \sqcap Q^{\cup} \circ S$$

e. $$\top\!\!\top \circ \ll \ =\ \top\!\!\top \circ \gg$$

(End of Axiom 3.9)
□

In the model, the projections are interpreted as:

$$f \,\langle\ll\rangle\, g \ \equiv\ \exists(\ h\ ::\ g = (f,h)\ )$$

$$f \,\langle\gg\rangle\, g \ \equiv\ \exists(\ h\ ::\ g = (h,f)\ )$$

A better-known interpretation of the projections is:

$$f \,\langle\ll\rangle\, (g,h) \ \equiv\ f = g$$

$$f \,\langle\gg\rangle\, (g,h) \ \equiv\ f = h$$

Notice, though, that this last definition does not cover the complete set of arguments: nothing is said about $f \,\langle\ll\rangle\, g$ where g cannot be split in two. It is understood that the interpretation only succeeds for pairs; otherwise the result is false.
With the interpretation in the model of $\ll$ and $\gg$ one can derive the interpretations of $\triangle$ and $\|$:

$$(f,g)\,\langle P \triangle Q\rangle\, h \ \equiv\ f\,\langle P\rangle\, h \wedge g\,\langle Q\rangle\, h$$

$$(f,g)\,\langle P \,\|\, Q\rangle\, (h,j) \ \equiv\ f\,\langle P\rangle\, h \wedge g\,\langle Q\rangle\, j$$

Later, in Section 4.4, a slightly different notation $f \blacktriangle g$ will be used for pairs (f,g). It would lead too far to explain the precise motivation already here.

The binary operators $\triangle$ and $\parallel$ have equal precedence, higher than all the other binary operators in the algebra. Parentheses will be used to disambiguate expressions where necessary.
Split and parallel composition are monotonic with respect to $\sqsubseteq$ because they are built using the monotonic operators $\circ$, $\sqcap$ and $^\cup$. Other consequences are the distribution of reverse over parallel composition, and the expressibility of the projections as a reversed split:

Theorem 3.11 *Parallel composition*

a. $(P \parallel Q)^\cup = P^\cup \parallel Q^\cup$

b. $\ll \;=\; (I \triangle \top\top)^\cup$

c. $\gg \;=\; (\top\top \triangle I)^\cup$

□

The following results can also be derived from the axioms. They are known as 'parallel-split fusion', 'parallel-parallel fusion' and 'computation rules'. The derivation involves elementary relational calculus, and illustrates the use of Axiom 3.10d:

Theorem 3.12 *Parallel-split fusion, parallel-parallel fusion*

a. $P \parallel Q \circ R \triangle S \;=\; (P \circ R) \triangle (Q \circ S)$

b. $P \parallel Q \circ R \parallel S \;=\; (P \circ R) \parallel (Q \circ S)$

Proof:

$P \parallel Q \circ R \triangle S$
$=$ { reverse is its own inverse }
$(P \parallel Q)^{\cup\cup} \circ R \triangle S$
$=$ { Theorem 3.11a }
$(P^\cup \parallel Q^\cup)^\cup \circ R \triangle S$
$=$ { Definition 3.10b }
$((P^\cup \circ \ll) \triangle (Q^\cup \circ \gg))^\cup \circ R \triangle S$
$=$ { Axiom 3.10d }
$(P^\cup \circ \ll)^\cup \circ R \sqcap (Q^\cup \circ \gg)^\cup \circ S$
$=$ { Axiom 3.8; reverse is its own inverse }
$\ll^\cup \circ P \circ R \sqcap \gg^\cup \circ Q \circ S$
$=$ { Definition 3.10a }
$(P \circ R) \triangle (Q \circ S)$

Theorem 3.12b follows from 3.12a by Definition 3.10b
□

For notational and calculational reasons, parallel composition is not associative. Also, the operator does not have a unit element. Notice that a choice for associativity of parallel composition would not have allowed the derived Theorem 3.12b. It can not be applied

to $P \parallel Q \circ R \parallel S \parallel T$, because it is not clear which parallel compositions match with each other. A similar remark holds for the absence of a unit element for parallel composition. Next, the computation rules for split are given:

Theorem 3.13 *Computation rules for split*

a. $\ll \circ P \triangle Q \;=\; P \sqcap \top\!\!\top \circ Q$

b. $\gg \circ P \triangle Q \;=\; Q \sqcap \top\!\!\top \circ P$

Proof:

$$
\begin{array}{ll}
 & \ll \circ P \triangle Q \\
= & \quad \{ \text{ Theorem 3.11b } \} \\
 & (I \triangle \top\!\!\top)^{\cup} \circ P \triangle Q \\
= & \quad \{ \text{ Axiom 3.10d } \} \\
 & I^{\cup} \circ P \sqcap \top\!\!\top^{\cup} \circ Q \\
= & \quad \{ \; I^{\cup} = I \text{; Axiom 3.8: } \top\!\!\top^{\cup} = \top\!\!\top \; \} \\
 & P \sqcap \top\!\!\top \circ Q
\end{array}
$$

□

In Subsection 3.2.1, the two expressions of Theorem 3.13 are further transformed to other, nicer expressions. Also, the computation rules for parallel composition are given in that subsection.
A thorough exploration of the properties of split and parallel composition can be found in Backhouse *et al.* [BdBH+91]. There are some minor differences in the notation, the main one being that we use the notation $\parallel$, whereas in [BdBH+91] the notation $\times$ is used.

3.1.5 Feedback

There are several ways to characterise the feedback operator. The one taken is the explicit formulation in terms of known procs:

Definition 3.14 *Feedback*

$$P^{\sigma} \;\triangleq\; (P \sqcap \gg) \circ I \triangle \top\!\!\top$$

□

Here, we use the meet with $\gg$ to model the feedback, whereas $I \triangle \top\!\!\top$ separates this feedback channel from the input. Notice that there are no type restrictions on P; the calculus gives as result that feedback only uses the binary input part of P. Using interpretations in the model given in the previous sections, the following interpretation for the feedback construction is derived:

$$f \langle P^{\sigma} \rangle g \equiv f \langle P \rangle (g, f)$$

The feedback construction obeys a rule of Input fusion:

Theorem 3.15 *Input fusion*

$$P^\sigma \circ Q \;=\; (P \circ Q \parallel I)^\sigma$$

□

Due to the fed-back output, it is not possible to find an equally short formulation for the form $P \circ Q^\sigma$. Because feedback is constructed from universally cupjunctive functions, it follows that feedback itself is universally cupjunctive. This is stated and proved in the next theorem.

Theorem 3.16 *Universal cupjunctivity*

$$(\sqcup(\, P \;:\; P \in \mathcal{B} \;:\; P\,))^\sigma \;=\; \sqcup(\, P \;:\; P \in \mathcal{B} \;:\; P^\sigma\,)$$

Proof:

$$\begin{array}{ll} & (\sqcup(\, P \;:\; P \in \mathcal{B} \;:\; P\,))^\sigma \\ = & \quad \{\ \text{Definition 3.14}\ \} \\ & (\sqcup(\, P \;:\; P \in \mathcal{B} \;:\; P\,) \sqcap \gg) \circ I \vartriangle \top\!\top \\ = & \quad \{\ \text{universal cupjunctivity of } \sqcap \gg \text{ and of } \circ\, I \vartriangle \top\!\top\ \} \\ & \sqcup(\, P \;:\; P \in \mathcal{B} \;:\; (P \sqcap \gg) \circ I \vartriangle \top\!\top\,) \\ = & \quad \{\ \text{Definition 3.14}\ \} \\ & \sqcup(\, P \;:\; P \in \mathcal{B} \;:\; P^\sigma\,) \end{array}$$

□

From the above it follows that the feedback operator $^\sigma$ is monotonic and $\perp\!\!\!\perp$-strict, i.e., $\perp\!\!\!\perp^\sigma = \perp\!\!\!\perp$. Another property is the computation rule for feedback:

$$P^\sigma \;=\; (P \sqcap \gg) \circ I \vartriangle P^\sigma$$

Notice the extra $\sqcap \gg$. In case proc P is *deterministic*, see Subsection 3.2.2, the rule can be simplified to:

$$P^\sigma \;=\; P \circ I \vartriangle P^\sigma$$

A derived feedback is a loop with a hidden feedback wire:

Definition 3.17 *Loop*

$$P^\varpi \;\triangleq\; \ll \circ\, (P \circ I \parallel \gg)^\sigma$$

□

With the interpretation in the model of the projections and compositions one can derive the interpretations of the loop:

$$f\,\langle P^\varpi \rangle\, g \;\equiv\; \exists(\, h \;::\; (f,h)\,\langle P \rangle\,(g,h)\,)$$

It turns out that the hidden feedback is as expressive as the feedback defined in Definition 3.14. The feedback construction of Definition 3.14 is obtained from Definition 3.17 by duplicating the output of the argument proc.

The construction obeys several fusion rules:

Theorem 3.18 *Feedback and loop*

a. $P^\sigma \;=\; (I \triangle I \circ P)^\varpi$

b. $P \circ Q^\varpi \circ R \;=\; (P \parallel I \circ Q \circ R \parallel I)^\varpi$

c. $(I \parallel P \circ Q)^\varpi \;=\; (Q \circ I \parallel P)^\varpi$

□

A corollary of Theorems 3.18a and 3.18b is $P \circ Q^\sigma \;=\; (P \triangle I \circ Q)^\varpi$. This looks like a good alternative for the dual of Theorem 3.15.
This concludes the introduction of the operators of the relational calculus. The following subsection will connect the layers of the algebra more tightly by adding two extra axioms.

3.1.6 The Cone Rule and Dedekind's Rule

An axiom that guarantees, among other consequences, that I and $\top\!\top$ differ from $\bot\!\bot$, is the Cone Rule. In general, the Cone Rule gives a condition to conclude that a proc is not equal to the uninteresting proc $\bot\!\bot$:

Axiom 3.19 *Cone Rule*

$$\top\!\top \circ P \circ \top\!\top \;=\; \top\!\top \;\equiv\; P \neq \bot\!\bot$$

□

Now, the claimed consequences can be proved:

Theorem 3.20 *Non-emptiness*

a. $I \neq \bot\!\bot$

b. $\top\!\top \neq \bot\!\bot$

c. $P \parallel Q = \bot\!\bot \;\equiv\; P = \bot\!\bot \vee Q = \bot\!\bot$

Proof:

$I \neq \bot\!\bot$
$=$ { Cone Rule 3.19 }
$\top\!\top \circ I \circ \top\!\top \;=\; \top\!\top$
$=$ { Axiom 3.3: I is identity of composition }
$\top\!\top \circ \top\!\top \;=\; \top\!\top$
$=$ { Theorem 3.6 }
True

The implication $\Leftarrow$ in 3.20c follows by $\bot\!\bot$-strictness of (the building blocks of) parallel composition.

For $\Rightarrow$ we assume $P \parallel Q = \bot\!\!\bot$ and $P \neq \bot\!\!\bot$:

$$
\begin{array}{ll}
 & P \parallel Q = \bot\!\!\bot \\
\Rightarrow & \quad \{ \ \bot\!\!\bot \text{ is zero of composition } \} \\
 & \gg \circ P \parallel Q \circ \top\!\!\top \vartriangle I \ = \ \bot\!\!\bot \\
= & \quad \{ \text{ Parallel-split fusion 3.12a } \} \\
 & \gg \circ \ (P \circ \top\!\!\top) \vartriangle Q \ = \ \bot\!\!\bot \\
= & \quad \{ \text{ Theorem 3.13b } \} \\
 & Q \sqcap \top\!\!\top \circ P \circ \top\!\!\top \ = \ \bot\!\!\bot \\
= & \quad \{ \ P \neq \bot\!\!\bot \text{: Cone Rule 3.19 } \} \\
 & Q \sqcap \top\!\!\top \ = \ \bot\!\!\bot \\
= & \quad \{ \ \top\!\!\top \text{ is unit of cap } \} \\
 & Q = \bot\!\!\bot
\end{array}
$$

□

A second axiom that connects the first three layers of the relational algebra is Dedekind's Rule. Although this rule looks terrifying (there are five free variables), it is very useful in calculations:

Axiom 3.21 *Dedekind's Rule*

$$
\begin{array}{lll}
 & P & \sqcap \ Q \circ R \ \sqsubseteq \ S \circ T \\
\Leftarrow & & \\
 & P \circ R^{\cup} \sqcap Q & \sqsubseteq \ S \\
\wedge & & \\
 & Q^{\cup} \circ P \quad \sqcap \quad R & \sqsubseteq \quad T
\end{array}
$$

□

To remember this rule one has to be able to reconstruct its syntactic shape. For example, the first conjunct of the antecedent is obtained as follows: take the consequent $P \sqcap Q \circ R \ \sqsubseteq \ S \circ T$, forget about the *right* arguments of the compositions (R and T) and add $R^{\cup}$ on the *right* side of P. The second conjunct is obtained in a symmetrical way.
In calculations it is often the case that one of the conjuncts becomes trivially true, for example if $Q = S$. So applying Dedekind does not really stretch the proofs.
A first formulation of Dedekind's Rule can be found in Riguet [Rig48]. Riguet attributes the rule to Dedekind who suggested a slightly weaker variation. Although the axiom only contains three free variables, it is less useful for our calculations, because it strongly tends to widen the proofs. It looks thus:

$$P \sqcap Q \circ R \ \sqsubseteq \ (P \circ R^{\cup} \sqcap Q) \ \circ \ (Q^{\cup} \circ P \sqcap R)$$

The interpretation in the model of Dedekind's rule, Axiom 3.21, boils down to the following statement in predicate calculus:

$$
\begin{array}{ll}
 & \forall (\ f, g, h \ : \ f \langle P \rangle g \wedge f \langle Q \rangle h \wedge h \langle R \rangle g \ : \ \exists (\ h' \ :: \ f \langle S \rangle h' \wedge h' \langle T \rangle g \ ) \ ) \\
\Leftarrow & \\
 & \forall (\ f, g, h \ : \ f \langle P \rangle g \wedge f \langle Q \rangle h \wedge h \langle R \rangle g \ : \ f \langle S \rangle h \wedge h \langle T \rangle g \ )
\end{array}
$$

In Backhouse *et al.* [BdBH+91] another rule is proposed: the so-called Middle Exchange Rule. It reads:

$$P^{\cup} \circ X \circ Q^{\cup} \sqsubseteq Y \;\equiv\; P \circ \neg Y \circ Q \sqsubseteq \neg X$$

In the scope of axioms stated earlier, the Middle Exchange Rule and Dedekind's Rule are equivalent. We prefer Dedekind's Rule because it lacks the negation.
To illustrate the use of Dedekind, consider the following theorem.

Theorem 3.22 *Swap*

$$P \sqcap Q \circ R \sqsubseteq \perp\!\!\!\perp \;\equiv\; P \circ R^{\cup} \sqcap Q \sqsubseteq \perp\!\!\!\perp$$

Proof by mutual implication:

$$\begin{array}{ll} & P \sqcap Q \circ R \sqsubseteq \perp\!\!\!\perp \\ = & \quad \{ \perp\!\!\!\perp \text{ is zero of composition } \} \\ & P \sqcap Q \circ R \sqsubseteq \perp\!\!\!\perp \circ R \\ \Leftarrow & \quad \{ \text{ Dedekind 3.21 } \} \\ & P \circ R^{\cup} \sqcap Q \sqsubseteq \perp\!\!\!\perp \\ = & \quad \{ \perp\!\!\!\perp \text{ is zero of composition } \} \\ & P \circ R^{\cup} \sqcap Q \sqsubseteq \perp\!\!\!\perp \circ R^{\cup} \\ \Leftarrow & \quad \{ \text{ Dedekind 3.21 } \} \\ & P \sqcap Q \circ R \sqsubseteq \perp\!\!\!\perp \end{array}$$

□

Another important theorem dealing with the distribution of sequential composition over cap is proved and some corollaries are stated. Notice that until now nothing has been said about such distributions.

Theorem 3.23 *Distribution over cap*

a. $(P \sqcap Q) \circ R \;=\; P \circ R \sqcap Q \circ R \;\Leftarrow\; P \circ R \circ R^{\cup} \sqsubseteq P$

b. $(P \sqcap Q \circ \top\!\!\top) \circ R \;=\; P \circ R \sqcap Q \circ \top\!\!\top$

Proof by mutual inclusion:
First:

$$\begin{array}{ll} & (P \sqcap Q) \circ R \sqsubseteq P \circ R \sqcap Q \circ R \\ = & \quad \{ \text{ plat calculus } \} \\ & (P \sqcap Q) \circ R \sqsubseteq P \circ R \;\wedge\; (P \sqcap Q) \circ R \sqsubseteq Q \circ R \\ \Leftarrow & \quad \{ \text{ monotonicity of composition } \} \\ & P \sqcap Q \sqsubseteq P \;\wedge\; P \sqcap Q \sqsubseteq Q \\ = & \quad \{ \text{ plat calculus } \} \\ & \mathit{True} \end{array}$$

And for the other inclusion:

$$
\begin{array}{ll}
 & P \circ R \sqcap Q \circ R \sqsubseteq (P \sqcap Q) \circ R \\
\Leftarrow & \quad \{ \text{ Dedekind 3.21 } \} \\
 & P \circ R \circ R^{\cup} \sqcap Q \sqsubseteq P \sqcap Q \\
\Leftarrow & \quad \{ \text{ plat calculus } \} \\
 & P \circ R \circ R^{\cup} \sqsubseteq P
\end{array}
$$

The proof of the second statement follows the same structure
□

Notice that the consequent of Theorem 3.23a is symmetric in P and Q. So in the antecedent, P could equally well have been replaced by Q. Some important corollaries with respect to distribution over split are:

Corollary 3.24 *Distribution over split*

a. $P \vartriangle Q \circ R = (P \circ R) \vartriangle (Q \circ R) \Leftarrow P \circ R \circ R^{\cup} \sqsubseteq P \vee Q \circ R \circ R^{\cup} \sqsubseteq Q$

b. $P \vartriangle (Q \circ \top\!\!\top) \circ R = (P \circ R) \vartriangle (Q \circ \top\!\!\top)$

□

This (almost) concludes the axiomatisation of the algebraic framework. In Section 3.4 a last axiom called *extensionality* will be added to the set of axioms for relational algebra $\mathcal{A}$. But first other important constructions have to be presented.

3.2 Definitions and properties

In this section, additional notions such as domains and functionality are defined and some of their most important properties are stated.

3.2.1 Interfaces, typing and domains

We are heading for some notion of *typing*. Take for example a relation R which relates elements of set A with elements of set B. The type of R can be represented by the inclusion $R \subseteq A \times B$. If the relation R is the square root function on $\mathbf{N}$ we could write in the model $\surd \subseteq \mathbf{R} \times \mathbf{N}$. An alternative notation is $\surd \in \mathbf{R} \leftarrow \mathbf{N}$. To capture the bi-direction which occurs in relations we propose for arbitrary proc P a more symmetric symbol: $P \in A \sim B$. In the following we will first concentrate on the notion of 'set A'; procs representing sets will be called *interfaces*. Then, the notation $P \in A \sim B$ can be introduced. Finally, the useful *domain operators*, which deliver the type of a proc, are defined.
Sets are represented as 'subsets' of the identity proc I, which represents the complete universe:

Definition 3.25 *Interfaces*

$$P \text{ is an interface } \equiv P \sqsubseteq I$$

□

Two trivial interfaces are the procs I and $\perp\!\!\!\perp$. Identifiers A and B are used to denote interfaces, that is, $A \sqsubseteq I$ and $B \sqsubseteq I$. In the model, interfaces can be interpreted by:

$$f \langle A \rangle g \Rightarrow f = g$$

These interfaces play the role of restricted identities. To stress the interpretation as a set, we write $f \in A$ for $f \langle A \rangle f$. A basic and useful theorem about interfaces is:

Theorem 3.26 *Interfaces*

a. $A = A^{\cup}$

b. $A \circ (P \sqcap Q) \;=\; A \circ P \sqcap Q$

□

The proof is omitted, though it is another good example of the usefulness of Dedekind's Rule 3.21. Notice that the second statement of Theorem 3.26 actually *equivales* the defining property of interfaces. This follows by instantiating P and Q both to I. Some straightforward corollaries of the theorem are:

Corollary 3.27 *Interfaces*

a. $A = A \circ A$

b. $A \circ B = B \circ A = A \sqcap B$

c. $A \circ (P \sqcap Q) \;=\; A \circ P \sqcap A \circ Q$

□

To introduce a more familiar way to express the type of a proc the following definition is given:

Definition 3.28 *Typing*

$$P \in A \sim B \;\equiv\; A \circ P = P \;\wedge\; P = P \circ B$$

□

In the model, the conjuncts $A \circ P = P$ and $P = P \circ B$ read:

$$f \langle P \rangle g \;\Rightarrow\; f \in A \wedge g \in B$$

This is equivalent to the set-theoretical notation $P \subseteq A \times B$. Other (different!) ways of typing could have been:

$$A \circ P = A \circ P \circ B = P \circ B$$

which is a weaker formulation than Definition 3.28. The disadvantage is that this definition only gives limited typing information: only in the presence of interface A can one remove B from $A \circ P \circ B$. An even weaker formulation is:

$$A \circ P \circ B = P \circ B$$

This alternative definition comes very close to conventional typing. It is, however, a highly asymmetric one, which is inconvenient in a framework which is closed under reverse, Axiom 3.8.
In the model it is easily seen that from $P \subseteq A \times B$, $A \subseteq C$ and $B \subseteq D$ it follows by monotonicity that P is also in $C \times D$. The question raised now is: given some proc P, what are the least A and B such that $P \subseteq A \times B$ is valid? These 'limit sets' seem to be useful, and therefore need to be defined in the formalism. They are called left domain ('range') and right domain ('source') and are denoted by $P<$ and $P>$, respectively:

Definition 3.29 *Domains*
Left domain:

$$P< \;\triangleq\; P \circ \top\!\!\top \sqcap I$$

Right domain:

$$P> \;\triangleq\; I \sqcap \top\!\!\top \circ P$$

□

From the definitions it follows that the domain operators are universally cupjunctive, which implies $\bot\!\!\bot< = \bot\!\!\bot> = \bot\!\!\bot$ and monotonicity of the operators. Notice that domains are interfaces. This is, among other useful properties, formally stated as follows:

Theorem 3.30 *Domains*

a. $P< \circ Q \;=\; P \circ \top\!\!\top \sqcap Q$ and $P \circ Q> \;=\; P \sqcap \top\!\!\top \circ Q$

b. $P< \circ P = P$ and $P = P \circ P>$

c. $P< \circ \top\!\!\top = P \circ \top\!\!\top$ and $\top\!\!\top \circ Q> = \top\!\!\top \circ Q$

d. $P< \sqsubseteq I$ and $P> \sqsubseteq I$

e. $A< = A> = A$

□

In the proof of Theorem 3.30, the distribution rule of Theorem 3.23b plays a fundamental role. The list of properties recorded in Theorem 3.30 is far from complete. During calculations in the chapters that follow, we will give extensive hints when dealing with those properties. It is a straightforward exercise to derive the interpretation of domains in the model. From Definition 3.29 one can conclude for the left domain:

$$f \in P< \;\equiv\; \exists(\, h \;::\; f \,\langle P \rangle\, h\,)$$

There are a number of equivalent formulations for typing a proc. Whenever appropriate, we feel free to take one of the equivalent formulations listed next.

Theorem 3.31 *Equivalent formulations for typing*
The following four statements are all equivalent:

a. $P = A \circ P \ \wedge \ P = P \circ B$

b. $P = A \circ P \circ B$

c. $P \sqsubseteq A \circ \top\!\top \ \wedge \ P \sqsubseteq \top\!\top \circ B$

d. $P{<} \sqsubseteq A \ \wedge \ P{>} \sqsubseteq B$

Proof by mutual implication:

$$
\begin{array}{ll}
 & P = A \circ P \ \wedge \ P = P \circ B \\
\Rightarrow & \quad \{ \text{ Leibniz } \} \\
 & P = A \circ P \circ B \\
\Rightarrow & \quad \{ \ \top\!\top \text{ is top element: monotonicity of composition } \} \\
 & P \sqsubseteq A \circ \top\!\top \ \wedge \ P \sqsubseteq \top\!\top \circ B \\
\Rightarrow & \quad \{ \text{ monotonicity of composition and cap; Theorem 3.6 } \} \\
 & P \circ \top\!\top \sqcap I \sqsubseteq A \circ \top\!\top \sqcap I \ \wedge \ I \sqcap \top\!\top \circ P \sqsubseteq I \sqcap \top\!\top \circ B \\
= & \quad \{ \text{ Definition 3.29; } A \text{ and } B \text{ are interfaces: Theorem 3.30e } \} \\
 & P{<} \sqsubseteq A \ \wedge \ P{>} \sqsubseteq B \\
\Rightarrow & \quad \{ \text{ monotonicity of composition } \} \\
 & P{<} \circ P \sqsubseteq A \circ P \ \wedge \ P \circ P{>} \sqsubseteq P \circ B \\
= & \quad \{ \text{ Theorem 3.30b } \} \\
 & P \sqsubseteq A \circ P \ \wedge \ P \sqsubseteq P \circ B \\
= & \quad \{ \ A \text{ is an interface: } P \sqsupseteq A \circ P \text{; anti-symmetry } \} \\
 & P = A \circ P \ \wedge \ P = P \circ B
\end{array}
$$

□

With the notion of domains, the computation rules for split, Theorem 3.13, can be rewritten, making use of Theorem 3.30a. Also the computation rules for parallel composition are given:

Theorem 3.32 *Computation rules for split and parallel composition, revised*

a. $\ll \circ \ P \vartriangle Q \ = \ P \circ Q{>}$

b. $\gg \circ \ P \vartriangle Q \ = \ Q \circ P{>}$

c. $\ll \circ \ P \parallel Q \ = \ P \circ \ll \circ \ I \parallel Q{>}$

d. $\gg \circ \ P \parallel Q \ = \ Q \circ \gg \circ \ P{>} \parallel I$

□

For a more intensive exploration of properties about typing and domains the reader is

referred to Backhouse *et al.* [BdBH+91]. In that paper, the term *monotypes* is used instead of interfaces.

3.2.2 Functionality and totality

Until now, nothing has been said about functions, functionality or determinism. All operators introduced thus far apply to arbitrary procs. These procs can be interpreted as (non-deterministic) relations. In this section it is specified what is meant by a functional (or deterministic) proc. Because injectivity is dual, it is defined simultaneously:

Definition 3.33 *Functionality, injectivity*

a. P is functional $\equiv P \circ P^{\cup} \sqsubseteq I$

b. P is injective $\equiv P^{\cup} \circ P \sqsubseteq I$

□

The identifiers F and G are used to denote arbitrary functional procs. As announced in the example $\surd \in \mathbf{R} \leftarrow \mathbf{N}$, the notation '$\leftarrow$' is used to denote that the proc is functional. So, by definition:

Definition 3.34 *Typing*

a. $P \in A \leftarrow B \equiv P \in A \sim B \wedge P$ is functional

b. $P \in A \rightarrow B \equiv P \in A \sim B \wedge P$ is injective

□

In the model the definition of functionality reads:

$$f \langle P \rangle h \wedge g \langle P \rangle h \Rightarrow f = g$$

For a function F, the formulation $f \langle F \rangle g$ can be interpreted as $f = F.g$, thus showing the embedding of functions in the algebra of relations. Notice that the direction of a proc 'from right to left' is implied. If one reads from left to right, the notion of functionality will become the notion of injectivity.
One of the main properties of functions is that they distribute from the right over cap and split. This follows from Theorem 3.23 and Corollary 3.24a:

Theorem 3.35 *Function distribution*

a. $(P \sqcap Q) \circ F = P \circ F \sqcap Q \circ F$

b. $F \circ (P \sqcap F^{\cup} \circ Q) = F \circ P \sqcap Q$

c. $P \triangle Q \circ F = (P \circ F) \triangle (Q \circ F)$

Proof:

$$
\begin{array}{ll}
 & (P \sqcap Q) \circ F \;=\; P \circ F \sqcap Q \circ F \\
\Leftarrow & \quad \{\ \text{Theorem 3.23a}\ \} \\
 & Q \sqsupseteq Q \circ F \circ F^{\cup} \\
\Leftarrow & \quad \{\ \text{monotonicity}\ \} \\
 & I \sqsupseteq F \circ F^{\cup} \\
= & \quad \{\ F \text{ ranges over functions; Definition 3.33a}\ \} \\
 & \mathit{True}
\end{array}
$$

And second, by mutual inclusion:

$$
\begin{array}{ll}
 & F \;\circ\; (P \sqcap F^{\cup} \circ Q) \\
\sqsubseteq & \quad \{\ \text{monotonicity of cap}\ \} \\
 & F \circ P \sqcap F \circ F^{\cup} \circ Q \\
\sqsubseteq & \quad \{\ F \text{ is a function: monotonicity}\ \} \\
 & F \circ P \sqcap Q \\
\sqsubseteq & \quad \{\ \text{Dedekind 3.21}\ \} \\
 & F \;\circ\; (P \sqcap F^{\cup} \circ Q)
\end{array}
$$

The proof of 3.35c is similar to the first proof and exploits Corollary 3.24a
□

Without proof we claim that Theorem 3.35a can be generalised to any (arbitrary) non-empty cap. The results of Theorem 3.35 are easily dualised to injections. Sequential composition, split and parallel composition all preserve functionality:

Theorem 3.36 *Preservation of functionality*

The composition operators $\circ$, $\|$ and $\vartriangle$ preserve functionality

□

The operators also preserve injectivity. The operator $\vartriangle$ preserves injectivity in a strong sense: injectivity of only one argument is required. This concludes the short discussion of functionality. Next, the notion of totality and the dual notion of surjectivity are defined:

Definition 3.37 *Totality, surjectivity*

a. P is total on $A \;\equiv\; A \sqsubseteq P{>}$

b. P is surjective to $A \;\equiv\; A \sqsubseteq P{<}$

□

For example, all procs are total on the interface $\perp\!\!\!\perp$. There are numerous equivalent formulations of totality. Two examples of formulations equivalent to 'P is total on A' are $A \sqsubseteq \top\!\!\top \circ P$ and $A \sqsubseteq P^{\cup} \circ P$. In the model, totality of P on A reads:

$$f \in A \;\Rightarrow\; \exists(\, h \;::\; h \,\langle P \rangle\, f\,)$$

The notation $\rightsquigarrow$ is used to denote that the proc is total; the reversed notation denotes surjectivity:

Definition 3.38 *Typing*

a. $P \in A \rightsquigarrow B \;\equiv\; P \in A \sim B \;\wedge\; P$ is total on B

b. $P \in A \leftsquigarrow B \;\equiv\; P \in A \sim B \;\wedge\; P$ is surjective to A

□

All combinations of 3.28, 3.34a, 3.34b, 3.38a and 3.38b are possible. The typings $\rightsquigarrow$ and $\leftarrow\!\!\!|$ are used very often. For example, $F \in \mathbf{N} \leftarrow\!\!\!| \mathbf{Z}$ expresses that F is a partial function, mapping *each* element in $\mathbf{Z}$ to a result in $\mathbf{N}$. For convenience, their precise meaning is written out in full:

Theorem 3.39 *Typing*

a. $P \in A \rightsquigarrow B \;\equiv\; P{<} \sqsubseteq A \;\wedge\; P{>} = B$

b. $P \in A \leftarrow\!\!\!| B \;\equiv\; P{<} \sqsubseteq A \;\wedge\; P{>} = B \;\wedge\; P$ is functional

Proof:

$P \in A \rightsquigarrow B$
$=$ { Definition 3.38a }
$P \in A \sim B \;\wedge\; P$ is total on B
$=$ { Definitions 3.28 and 3.37a }
$A \circ P = P \;\wedge\; P = P \circ B \;\wedge\; B \sqsubseteq P{>}$
$=$ { equivalence of 3.31a and 3.31d }
$P{<} \sqsubseteq A \wedge P{>} \sqsubseteq B \wedge B \sqsubseteq P{>}$
$=$ { anti-symmetry }
$P{<} \sqsubseteq A \wedge P{>} = B$

□

With these definitions of interfaces and domains, functionality and injectivity, and totality and surjectivity one can list some typing properties of the basic procs. For example, the property that all interfaces are isomorphisms on 'themselves' is stated:

Theorem 3.40 *Typing of basic procs*
For all interfaces A:

a. $A \in A \leftrightarrow\!\!\!\!| A$

In particular:

b. $\perp\!\!\!\perp \in \perp\!\!\!\perp \leftrightarrow\!\!\!\!| \perp\!\!\!\perp$

c. $I \in I \leftrightarrow\!\!\!\!| I$

Moreover:

d. $\top\!\!\top \in I \sim\!\!\!\rhd I$

e. $\ll\ \in\ I \hookleftarrow I \parallel I$

f. $\gg\ \in\ I \hookleftarrow I \parallel I$

□

The top relation $\top\!\!\top$ is the most non-deterministic proc one can imagine; so it is not functional. Because $\top\!\!\top$ is symmetric, $\top\!\!\top^{\cup} = \top\!\!\top$, injectivity is also ruled out. The projections destruct their argument: picking out one element from a pair, forgetting the other one. Therefore, projections can not be injective. This concludes the limited discussion on the type information of procs. In Section 3.4 we will introduce some special functions called *points*.
The following theorem summarises preservation properties of fundamental composition operators:

Theorem 3.41 *Preservation of typing*

a. $P \sqcup Q\ \in\ A \sqcup B \sim C \sqcup D\ \ \Leftarrow\ \ P \in A \sim C\ \ \wedge\ \ Q \in B \sim D$

b. $P \circ Q \in A \sim C\ \ \Leftarrow\ \ P \in A \sim B\ \ \wedge\ \ Q \in B \sim C$

c. $P \vartriangle Q\ \in\ A \parallel B \sim C \sqcap D\ \ \Leftarrow\ \ P \in A \sim C\ \ \wedge\ \ Q \in B \sim D$

d. $P \parallel Q\ \in\ A \parallel B \sim C \parallel D\ \ \Leftarrow\ \ P \in A \sim C\ \ \wedge\ \ Q \in B \sim D$

□

The proof is omitted. This concludes the limited discussion on the type information of procs.

3.3 Demonic composition

This section introduces demonic sequential composition. Recall that the composition which has been used until now is angelic relational composition. It is not our objective to introduce demonic composition in yet another way, or to investigate it thoroughly; other authors have done that before, and well enough, see Backhouse & van der Woude [BvdW93], Berghammer [Ber91] or Sekerinski [Sek93].
Sometimes, we will 'step outside' our formalism and look at some operator from the point of implementation. In this way several choices are motivated.
In developing a calculus for the design of (distributed) algorithms the task is to give a set of rules and to provide programming heuristics as general and powerful as possible. In this way, the final calculus will be widely applicable and easy to use. This is the motivation to choose an axiomatic introduction of an algebra which contains several *angelic* operators.

The present calculus supplies a sequential composition $\circ$ and a union $\sqcup$ which are combined in an axiom stating the universal distribution over union of composition in both its arguments. This axiom results in many nice and easy to remember operators and calculation rules such as the existence of the factors $\backslash$ and $/$, and monotonicity and $\bot\!\!\!\bot$-strictness of composition.

3.3.1 Demonic composition defined

Looking at (hardware) implementations, components are connected by wires. What is the theoretical counterpart of connecting components sequentially in our algebra? At first glance it looks as if sequential composition $\circ$ is the right one.
But this composition is an angelic operator in the sense that $P \circ Q$ relates an input to an output whenever there exists *some* intermediate result between P and Q:

$$\begin{array}{ll} & f \,\langle P \circ Q \rangle\, g \\ \equiv & \\ & \exists(\, h \;::\; f \,\langle P \rangle\, h \wedge h \,\langle Q \rangle\, g \,) \end{array}$$

This does not behave like the connection of P and Q obtained by a wire because the output of proc Q depends on the (in-)ability of proc P to process this output. We need a connection of P and Q that makes it possible for component Q to produce output regardless of whether the following component P can consume it. In that case, it will be possible to implement the sequential composition of P and Q by using the individual implementations of P and Q (*compositionality*: the meaning of $P \otimes Q$ is a function of the meaning of P and the meaning of Q). The connection we are heading for is known as demonic composition.
As a side-remark we mention that there are already several demonic-like operators present in the algebra: among others, there is split and parallel composition.
Motivated by the informal behaviour of a 'wire' we propose a more stringent composition, dubbed demonic composition: $P \diamond Q$ behaves like the relational composition restricted to those initial inputs which only lead to intermediate values for which P is defined:

$$\begin{array}{ll} & f \,\langle P \diamond Q \rangle\, g \\ \equiv & \\ & f \,\langle P \circ Q \rangle\, g \wedge \forall(\, h \;:\; h \,\langle Q \rangle\, g \;:\; h \in P_{>} \,) \end{array}$$

In the literature, various ways to construct a point-free formulation for demonic composition are presented. We just jump to the definition with only the remark that the $\forall$-quantification corresponds to a factor $/$ in the relational calculus:

Definition 3.42 *Demonic composition*

$$P \diamond Q \;\triangleq\; P \circ Q \;\sqcap\; (\top\!\!\top \circ P) \,/\, Q^{\cup}$$

□

This definition of demonic composition is the same as the one given by Backhouse & van der Woude [BvdW93]. Demonic composition has the same priority as angelic compo-

sition. It is well known that demonic composition is a monoid operation: it is associative and has I as identity:

Theorem 3.43 *Demonic composition*

$(\mathcal{A}, \mathbin{⍟}, I)$ is a monoid

□

This is a nice property of demonic composition. However, a major disadvantage of demonic composition is the fact that it is only monotonic with respect to the order $\sqsubseteq$ on procs in its first argument P. It is neither monotonic nor anti-monotonic in Q. This rules out general junctivity properties.
The thing we are interested in for reasons of refinement and, even more importantly, implementation, is conditions under which our well-behaved angelic composition coincides with the newly introduced demonic composition. The next theorem states some useful preliminary results:

Theorem 3.44 *Demonic composition*

a. $P \circ Q = P \mathbin{⍟} Q \;\equiv\; P \circ Q \circ Q^{\cup} \sqsubseteq \top\!\top \circ P$

b. $(P \mathbin{⍟} Q)^{>} = Q^{>} \;\wedge\; P \circ Q = P \mathbin{⍟} Q \;\equiv\; Q^{<} \sqsubseteq P^{>}$

Proof:

$$
\begin{array}{ll}
 & P \circ Q = P \mathbin{⍟} Q \\
= & \quad \{ \text{ Definition 3.42 } \} \\
 & P \circ Q \;=\; P \circ Q \sqcap (\top\!\top \circ P) \,/\, Q^{\cup} \\
= & \quad \{ \text{ plat calculus } \} \\
 & P \circ Q \sqsubseteq (\top\!\top \circ P) \,/\, Q^{\cup} \\
= & \quad \{ \text{ Definition 3.4b } \} \\
 & P \circ Q \circ Q^{\cup} \sqsubseteq \top\!\top \circ P
\end{array}
$$

In the proof of the second statement, many rules from the domain calculus are used:

$$
\begin{array}{ll}
 & (P \mathbin{⍟} Q)^{>} = Q^{>} \;\wedge\; P \circ Q = P \mathbin{⍟} Q \\
= & \quad \{ \text{ Leibniz; Theorem 3.44a } \} \\
 & (P \circ Q)^{>} = Q^{>} \;\wedge\; P \circ Q \circ Q^{\cup} \sqsubseteq \top\!\top \circ P \\
= & \quad \{ \text{ Theorem 3.30c; calculus } \} \\
 & \top\!\top \circ P \circ Q = \top\!\top \circ Q \;\wedge\; \top\!\top \circ P \circ Q \circ Q^{\cup} \sqsubseteq \top\!\top \circ P \\
= & \quad \{ \text{ Leibniz } \} \\
 & \top\!\top \circ P \circ Q = \top\!\top \circ Q \;\wedge\; \top\!\top \circ Q \circ Q^{\cup} \sqsubseteq \top\!\top \circ P \\
= & \quad \{ \text{ monotonicity; domain calculus: } \top\!\top \circ Q \circ Q^{\cup} = \top\!\top \circ Q^{\cup} \; \} \\
 & \top\!\top \circ P \circ Q \sqsupseteq \top\!\top \circ Q \;\wedge\; \top\!\top \circ Q^{\cup} \sqsubseteq \top\!\top \circ P \\
= & \quad \{ \; \top\!\top \circ Q^{\cup} \sqsubseteq \top\!\top \circ P \;\text{ implies }\; \top\!\top \circ Q = \top\!\top \circ Q^{\cup} \circ Q \sqsubseteq \top\!\top \circ P \circ Q \; \} \\
 & \top\!\top \circ Q^{\cup} \sqsubseteq \top\!\top \circ P \\
= & \quad \{ \text{ Theorem 3.30c } \} \\
 & Q^{<} \sqsubseteq P^{>}
\end{array}
$$

□

The main corollary of Theorem 3.44b is:

Corollary 3.45 *Demonic composition*

a. $P \circ Q = P \mathbin{⨾} Q \;\Leftarrow\; Q^{<} \sqsubseteq P^{>}$

b. $P \circ Q = P \mathbin{⨾} (P^{>} \circ Q)$

c. $P \circ Q = P \mathbin{⨾} Q \;\Leftarrow\; Q$ is a function

Proof:
Statement 3.45a follows from Theorem 3.44b. The second one 3.45b is just a concise rewriting of 3.45a. For 3.45c we calculate:

$$
\begin{array}{ll}
 & P \circ Q = P \mathbin{⨾} Q \\
= & \quad \{ \text{ Theorem 3.44a } \} \\
 & P \circ Q \circ Q^{\cup} \sqsubseteq \top\!\top \circ P \\
\Leftarrow & \quad \{ \text{ monotonicity } \} \\
 & P \sqsubseteq \top\!\top \circ P \;\wedge\; Q \circ Q^{\cup} \sqsubseteq I \\
= & \quad \{ \; I \sqsubseteq \top\!\top \text{: monotonicity of composition } \} \\
 & Q \circ Q^{\cup} \sqsubseteq I \\
= & \quad \{ \text{ Definition 3.33a } \} \\
 & Q \text{ is a function}
\end{array}
$$

□

This corollary justifies the remark that angelic composition and demonic composition coincide whenever the procs are total. What is actually meant is that every proc has to be total on the left domain of the proceeding proc.
The condition for the equality we derived in Theorem 3.44a can be interpreted in the model as follows:

$$
\begin{array}{ll}
 & P \circ Q \circ Q^{\cup} \sqsubseteq \top\!\top \circ P \\
= & \\
 & \forall(\, f, g \;:\; f \,\langle Q \circ Q^{\cup} \rangle\, g \;:\; f \in P^{>} \Rightarrow g \in P^{>} \,) \\
= & \\
 & \forall(\, f, g, h \;:\; f \,\langle Q \rangle\, h \wedge g \,\langle Q \rangle\, h \;:\; f \in P^{>} \equiv g \in P^{>} \,)
\end{array}
$$

In words this means: any two outputs f and g generated (via Q) by one input h are either both inside the domain of P or both outside that domain.
Corollary 3.45c concluded that functionality of Q is also sufficient to establish the equality $P \circ Q = P \mathbin{⨾} Q$. One could argue that functionality is also desirable in programs. Therefore, to refine a specification down to some demonic composition of procs, it suffices to end up with just functions. However, in that case we might loose totality: we can not guarantee that the refinement we found is as total as its 'first' component. For example, in the above formula, we would like to conclude that $(P \mathbin{⨾} Q)^{>} = Q^{>}$. For that to follow we need type information about the sequential connections; type information like $Q^{<} \sqsubseteq P^{>}$. So, we need the information anyway.

In calculations, we will continue to use angelic composition. However, we feel free to impose type assumptions (according to Corollary 3.45a) when needed.
This subsection introduced the demonic composition in an angelic calculus. In this way, we preserved all nice calculational properties of the angelic relational algebra. The conclusion is that an angelic environment is well-suited to define and investigate demonic operators. As a result, we obtain a calculus in which angelic and demonic operators are at the disposal of the program designer.

3.3.2 Weakest preconditions

There is a nice way to relate the demonic composition to the well-known sequential composition ; in imperative programming. The connection is given via *weakest preconditions*. In Dijkstra & Scholten [DS90], and in many other books on program semantics, the semantics of ; is given by:

$$wp.(S_0\,;S_1).X = wp.S_0.(wp.S_1.X)$$

where S_0 and S_1 are statements, and X ranges over predicates. This rule expresses the behaviour of the compound statement $S_0\,;S_1$ as a combination of the individual behaviours of S_0 and S_1, i.e., compositionally.
To make a connection between demonic composition $\natural$ and this wp-rule we first identify predicates and the predicate transformer $wp.P$. Let the predicates be modelled by interfaces, i.e., p is a predicate if and only if $p \sqsubseteq I$. Recall that procs under I represent *sets* and that predicates and sets are in one-to-one correspondence. The predicates form a complete, universally distributive (complemented) lattice. The meet is $\sqcap$ and will be denoted (very suggestively) by $\wedge$. Consequently, the inclusion $\sqsubseteq$ on predicates can be denoted by $\Rightarrow$, and equality $=$ becomes equivalence $\equiv$.
The predicate transformer $wp.P$ is defined for all procs P:

Definition 3.46 *Weakest precondition*

$$wp.P.p \triangleq (p \natural P)\rangle$$

□

Others, like Doornbos [Doo94] and Sekerinski [Sek93], already suggested this connection between demonic composition and wp in various ways. In the model, the weakest precondition of P for postcondition p reads:

$$\begin{array}{ll} & f \in wp.P.p \\ \equiv & \\ & \exists(\, g \;:\; g\,\langle P\rangle\, f \;:\; g \in p\,) \wedge \forall(\, g \;:\; g\,\langle P\rangle\, f \;:\; g \in p\,) \end{array}$$

This is the interpretation: $wp.P.p$ contains exactly those input chronicles for which proc P is guaranteed to result in an output chronicle in p.

The compound statements demonic sequential composition $\natural$, split $\triangle$ and parallel composition $\parallel$ enjoy the following *wp* properties:

$$wp.(P \natural Q).p \equiv wp.Q.(wp.P.p)$$

$$wp.(P \triangle Q).(p \parallel q) \equiv wp.P.p \wedge wp.Q.q$$

$$wp.(P \parallel Q).(p \parallel q) \equiv wp.P.p \parallel wp.Q.q$$

As an example, we will prove the *wp*-rule for demonic composition. A nice tool is the demonic counterpart of $(P \circ Q)_> = (P_> \circ Q)_>$. It reads $(P \natural Q)_> = (P_> \natural Q)_>$ and is derived mainly using $\top\!\!\top_> = I$ and Corollary 3.45a:

$$\begin{array}{ll} & wp.(P \natural Q).p \\ = & \quad \{ \text{ Definition 3.46 } \} \\ & (p \natural P \natural Q)_> \\ = & \quad \{ \ (P \natural Q)_> = (P_> \natural Q)_> \ \} \\ & ((p \natural P)_> \natural Q)_> \\ = & \quad \{ \text{ Definition 3.46 } \} \\ & (wp.P.p \natural Q)_> \\ = & \quad \{ \text{ Definition 3.46 } \} \\ & wp.Q.(wp.P.p) \end{array}$$

The second and third property use predicates $p \parallel q$; these are predicates on pairs. The third property in particular shows the absence of communication between P and Q.
By Corollary 3.45b it follows that $wp.(P \circ Q).p \equiv wp.(P_> \circ Q).(wp.P.p)$ which once more reflects that the behaviour of Q is restricted by (the right domain of) P.
This concludes the short discussion of the demonic composition $\natural$. In the next section we will introduce some special functions called *points*.

3.4 Points and extensionality

In this section we define *points*. Points are constant total functional procs. They represent the elements f of the model, and enable the formulation of another axiom called the *Principle of Extensionality* in algebra $\mathcal{A}$:

Definition 3.47 *Points*

$$\begin{array}{ll} & point.x \\ \equiv & \\ & x = x \circ \top\!\!\top \ \wedge \ x \in I \leftrightarrow I \end{array}$$

□

We let identifiers x and y range over points. In the model the conjunct $x = x \circ \top\!\!\top$ is interpreted as:

$$f \langle x \rangle g \equiv f \langle x \rangle h$$

It represents the 'constant' aspect of points. In words it says: no matter what the input (g or h), the output is always the same (f). Some properties of points used in the sequel are:

Theorem 3.48 *Points*

a. $x \circ x^{\cup} = x^{<}$

b. $x \mathbin{\triangle} I \circ P = x \mathbin{\triangle} P$

c. $\top\!\top \circ x = \top\!\top$

□

The first statement follows from the functionality of points and is in fact a property that is valid for all functions ($P \circ P^{\cup} = P^{<}$ is even equivalent to stating that P is a function). The second follows from the property that points are constant. The last property describes totality on I.
With points it is possible to formulate and axiomatise the Principle of Extensionality. In set theory, this principle states that two sets are equal if and only if they contain the same elements. In functional programming the mechanism of extensionality is used to prove that two functions are equal:

$$f = g \;\equiv\; \forall(\, x \;::\; f.x = g.x \,)$$

This is easily generalised to relations, lifting *function application* to *relational composition*. In the relational algebra the above translates to the axiom:

Axiom 3.49 *Extensionality*

$$P = Q \;\equiv\; \forall(\, x \;::\; P \circ x = Q \circ x \,)$$

□

This is the last axiom for the relational algebra $\mathcal{A}$. No more axioms on the algebra are needed to prove the desired results in the algebra of procs. There are many other possible formulations of extensionality, some of which are:

Theorem 3.50 *Equivalent formulations of Extensionality*

a. $I = \sqcup(\, x \;::\; x \circ x^{\cup} \,)$

b. $\top\!\top = \sqcup(\, x \;::\; x \,)$

□

For an intensive exploration of points and extensionality in the relational calculus the reader is referred to Rietman [Rie91]. It is the objective to try to avoid the use of extensionality as much as possible, and do the calculations at the level of procs: they are our first-class citizens.

3.5 Two preservation problems

A healthiness condition of a relational calculus for the design of distributed algorithms should be that it subsumes the results of Kahn for deterministic procs, Kahn [Kah74]. One property of the feedback operator should be its preservation of functionality. This is, in general, a false statement in the calculus introduced thus far. This is shown by the following example: Theorem 3.40e states that the proc $\gg$ is deterministic, which can not be said of the proc $\gg^{\sigma}$:

$$
\begin{array}{ll}
 & \gg^{\sigma} \\
= & \quad \{ \text{ Definition Feedback 3.14 } \} \\
 & (\gg \sqcap \gg) \circ I \vartriangle \top\!\!\top \\
= & \quad \{ \text{ idempotency of } \sqcap \ \} \\
 & \gg \circ I \vartriangle \top\!\!\top \\
= & \quad \{ \text{ Split computation 3.32 } \} \\
 & \top\!\!\top \circ I_{>} \\
= & \quad \{ \ I_{>} = I; \text{ identity } \} \\
 & \top\!\!\top
\end{array}
$$

We can not deduce from the relational calculus that $\top\!\!\top$ is deterministic; even stronger: in the model for distributed programming we are heading for, the proc $\top\!\!\top$ is *not* deterministic. Our conclusion is that in general feedback does not preserve determinism.
The question raised now is: do we want our feedback operator to preserve determinism? In general, the answer to the question is 'no'. The result derived above is actually what we want. Still, we might be tempted to say that realistic procs which are deterministic do not behave non-deterministically when they are put in a feedback loop. The projection proc in the example above looks ordinary enough to be entitled a realistic proc.
The problem is, though, that the projection is *not* realistic since it exhibits instantaneous response, which is not a realistic behaviour: all realistic deterministic procs suffer from some delay in their response.
In Section 3.3 we motivated why demonic composition is required and how to transform angelic composition into the corresponding demonic counterpart: Corollary 3.45a. This result points out that the type of a proc is important. Therefore, we need a type inference rule for the operators of the calculus. The operators cup, sequential composition, split and parallel composition do not cause any problem, see Theorem 3.41. However, the feedback operator does cause serious problems (because cap does). We do not have some rule like:

$$
\begin{array}{ll}
 & P^{\sigma} \in A \rightsquigarrow B \\
\Leftarrow & \\
 & P \ \in \ A \rightsquigarrow B \parallel A
\end{array}
$$

This is one of the reasons why we did not give a *wp*-rule for feedback in Subsection 3.3.2. So, we found a second problem of feedback: typing is not preserved. After having introduced time in the calculus and a few other constructions, we can tackle the preservation properties of feedback in Chapter 7.

Brock and Ackerman point out in their paper [BA81] that the history model of deterministic networks, where procs are functions between input and output histories, cannot be extended to a history model of non-deterministic networks, where procs are relations. To solve the problems they introduce a network model they called *scenarios*. This model is the history model of non-deterministic networks restricted by a causality requirement on the input and output histories. This causality requirement captures information of timing aspects between input and output.
In the spirit of Brock and Ackerman these problems are solved by adding time information. The introduction of time in the model is the main subject of Chapter 4. When the two problems are solved, making use of the time information, we can abstract from time in the model by axiomatising the results in the calculus.

Chapter 4

Back to the model

To be able to describe causality properties in our set-theoretical model for binary relations, the notion of *time* is introduced. This guides the way to the solution of the problems about the functionality preservation and the totality preservation of feedback.
The universe we are interested in is $\mathcal{C}$, which is a subset of $\mathcal{M} \leftarrow\!\!\!\!\!\!\rightarrow \mathcal{T}$. The elements of $\mathcal{C}$ are called *chronicles*. The type $\mathcal{M}$ is some message domain; $\mathcal{T}$ is some time domain. The time domain and the exact definition of the set of chronicles will be the main subject of discussion in this chapter.
Chronicles describe a complete timed (input or output) history. In Chapter 2 we argued why we restricted our attention to binary relations. In operational terms the notation $f \langle P \rangle g$ reads: for input g a possible output of P is f. Now we know what the interpretations of g and f are, we can give the complete interpretation of $f \langle P \rangle g$: for some input chronicle g, which describes the input for every moment in time, a possible output chronicle is f, describing again the output for every moment in time.
The chronicles are taken from a relational algebra $\mathcal{B}$. The reason for this choice is that it enables (forces) us to do the calculations (in the model) precisely and formally. Section 4.1 introduces the algebra $\mathcal{B}$ after which the axiomatisation of the message domain and the time domain $\mathcal{T}$ follows in Sections 4.2 and 4.3; in those sections, two additional assumptions are imposed on $\mathcal{B}$. Then, the set of chronicles is defined in Section 4.4. Finally, the Axiom of Choice for Chronicles is clarified and imposed on algebra $\mathcal{B}$ in Section 4.5.

4.1 Relational algebra $\mathcal{B}$

Assume a relational algebra $(\mathcal{B}, \subseteq, \neg, \bullet, \mathcal{I}, ^{-1}, \ll, \gg)$ satisfying the axioms of a relational algebra as introduced in the Chapter 3. The Principle of Extensionality is assumed to be valid in $\mathcal{B}$.
The elements of $\mathcal{B}$ are called relations; the identifiers r and s range over $\mathcal{B}$. The join and meet compositions in $\mathcal{B}$ are denoted by $\cup$ and $\cap$, respectively, with unit elements $\bot$ and $\top$. For split and parallel composition in $\mathcal{B}$ the notations $\blacktriangle$ and $\times$ are used; for left and right domains in $\mathcal{B}$ the notations $\prec$ and $\succ$ are used. Typing of relations in $\mathcal{B}$ will be denoted the

same as the typing of procs in $\mathcal{A}$. Relations below $\mathcal{I}$ are called *identities*. The identifiers a and b range over these identity relations. Without further comments, all results for $\mathcal{A}$ stated in Chapter 3 will be used for $\mathcal{B}$ also.

4.2 Message domain

First, the message domain is described in more detail. To be able to reason about *a single channel*, the identity on 'non-pairs' is defined. We simply take the polymorphic identity $\mathcal{I}$ and exclude all pairs:

Definition 4.1 *Singleton identity*

$$\iota \triangleq \mathcal{I} \cap \neg(\mathcal{I} \times \mathcal{I})$$

□

This definition implies that $\mathcal{I}$ solves the equation $x :: x = \iota \cup x \times x$. To ensure that ι is non-empty, we assume that $\mathcal{I}$ is the least solution of the equation. If ι were $\bot$, $\mathcal{I}$ would be $\bot$ which contradicts the Cone Rule in $\mathcal{B}$. It is assumed that there is a special singleton message nm in ι to formalise the notion of 'no message':

Assumption 4.2 *non-empty ι spans $\mathcal{I}$*

a. $\mathcal{I} = \mu(\, x :: \iota \cup x \times x \,)$

b. $nm \neq \bot \;\wedge\; nm \bullet \top \bullet nm \subseteq \iota$

□

The complement of nothing with respect to the identity ι is msg, i.e., the identity on *messages* on a single channel:

Definition 4.3 *Singleton message*

$$msg \triangleq \neg nm \cap \iota$$

□

To decide whether there is a useful message somewhere (possibly within a nested pair), we introduce the identity sm:

Definition 4.4 *Somewhere a message*

$$sm \triangleq \mu(\, x :: msg \cup x \times \mathcal{I} \cup \mathcal{I} \times x \,)$$

□

The relation sm is an identity on singletons or pairs (or pairs of pairs, etc.) that carry a useful message somewhere.

4.3 Time domain

This section defines the time domain $\mathcal{T} \in \mathcal{B}$. A possible model we have in mind for $\mathcal{T}$ is the set of reals $\mathbf{R}$. We will, however, only use properties of $\mathcal{T}$ as an ordered set in the assumptions, and thus provide no means for obtaining a metric on $\mathcal{T}$. If a metric is desired, for example for a real-time calculus, additional assumptions must be imposed.
Recall that a partially ordered set $(\mathcal{S}, \preceq)$ is a chain if any two elements of the set $\mathcal{S}$ are comparable, see Davey & Priestley [DP90]. We call a chain $(\mathcal{S}, \preceq)$ *closed and infinite* if $\mathcal{S}$ has all limits (least upper bounds) of bounded increasing sequences, and if the relation $\prec$ is total and surjective on $\mathcal{S}$. The transitive and irreflexive relation $\prec$ is induced by $\preceq$. Totality and surjectivity of $\prec$ excludes minimal and maximal elements, and therefore $(\mathcal{S}, \preceq)$ is called infinite. Examples of closed infinite chains are $\mathbf{Z}$ and $\mathbf{R}$ under the order $\leqslant$. The rationals $\mathbf{Q}$ do not form a closed infinite chain because $\mathbf{Q}$ does not have all limits of bounded increasing sequences. Next, we assume the existence of the time domain $\mathcal{T}$:

Assumption 4.5 *Time domain $\mathcal{T}$*

> The time domain $\mathcal{T}$ is a non-empty identity of $\mathcal{B}$ with a partial order $\leqslant$ such that $(\mathcal{T}, \leqslant)$ is a closed infinite chain

□

In Chapter 10, another (optional) assumption will be imposed on $\mathcal{T}$ making the time domain isomorphic to the set of reals $\mathbf{R}$. We will keep track of all the consequences of this assumption, thereby tagging the theorems of the calculus which are only valid for a continuous time domain.
The induced strict relation $<$ and their reversed versions $\geqslant$ and $>$, defined in the usual way, are used on $\mathcal{T}$. For reference, we record a few propositions:

Proposition 4.6
For all relations $r \in \mathcal{T} \sim \mathcal{T}$ and s:

$$r \subseteq s \cup \leqslant \;\equiv\; r \cap > \subseteq s$$

□

Proposition 4.6 is equivalent to Proposition 4.7. The proposition states that $\leqslant$ and $>$ are complementary relations with respect to $\mathcal{T} \bullet \top \bullet \mathcal{T}$:

Proposition 4.7

a. $\leqslant \cap > \;=\; \bot$

b. $\leqslant \cup > \;=\; \mathcal{T} \bullet \top \bullet \mathcal{T}$

□

Writing out the details, this boils down to saying that $<$, $\mathcal{T}$ and $>$ establish a trichotomy on $\mathcal{T}$.

The ordering $\leqslant$ cancels out when composed with $<$:

Proposition 4.8

$$< \bullet \leqslant \;=\; < \;=\; \leqslant \bullet <$$

Proof:

$$\begin{array}{ll} & < \bullet \leqslant \\ = & \quad \{ \text{ Assumption 4.5: } \leqslant \;=\; < \cup \mathcal{T} \ \} \\ & < \bullet (< \cup \mathcal{T}) \\ = & \quad \{ \text{ cupjunctivity of composition } \} \\ & < \bullet < \cup < \bullet \mathcal{T} \\ = & \quad \{ \text{ Assumption 4.5: } < \text{ is total on } \mathcal{T} \ \} \\ & < \bullet < \cup < \\ = & \quad \{ \ < \text{ is transitive: plat calculus } \} \\ & < \end{array}$$

□

To be able to reason about *elements* of $\mathcal{T}$ we introduce moments in time t. Their properties in $\mathcal{B}$ are defined as follows:

Characterisation 4.9 *Moment*

$$\begin{array}{ll} & t \text{ is a moment} \\ \equiv & \\ & t \neq \bot \ \wedge \ t \bullet \top \bullet t \subseteq \mathcal{T} \end{array}$$

□

The existence of moments is guaranteed by the Principle of Extensionality in $\mathcal{B}$. Some properties for these moments are:

Proposition 4.10

a. $t \subseteq \mathcal{T}$

b. $t \bullet \top \bullet t = t$

c. $t \bullet \top \in t \mathbin{\kappa\!\!\!\mapsto} \mathcal{I}$

□

For an extensive exploration of relations satisfying Characterisation 4.9, one is referred to Backhouse *et al.* [BdBH+91]. In that paper, the relations are dubbed 'unit types' and are below I.
A corollary of Proposition 4.10c is that $t \bullet \top$ is a point below $\mathcal{T}$. This allows extensional arguments over moments, for example $\mathcal{T} = \cup(t \ :: \ t)$.

4.4 Chronicles

This section introduces the chronicles which are used to describe the stream of messages on a channel. The set of chronicles $\mathcal{C}$ is a subset of the set of relations $\mathcal{B}$. There will be some limitations on the message domain of chronicles to assure the well-definedness of their arity. Also, the time domain will be restricted to guarantee an induction principle.
With sm, Definition 4.4, we can define the support sp of a relation total on $\mathcal{T}$. For relation $r \in \mathcal{I} \rightsquigarrow \mathcal{T}$, the identity $sp.r$ is a subset of $\mathcal{T}$ containing those moments where relation r really has something to tell, i.e., is able to return a useful message, possibly in a pair:

Characterisation 4.11 *Support*
For $r \in \mathcal{I} \rightsquigarrow \mathcal{T}$:

$$sp.r \triangleq (sm \bullet r)_\succ$$

□

It is time to investigate a proposition which states that the function sp distributes over the split $\blacktriangle$:

Proposition 4.12

$$sp.(r \blacktriangle s) = sp.r \cup sp.s$$

Proof:

$$\begin{array}{ll} & sp.(r \blacktriangle s) \\ = & \quad \{ \text{ Characterisation 4.11 } \} \\ & (sm \bullet r \blacktriangle s)_\succ \\ = & \quad \{ \text{ Definition 4.4 } \} \\ & ((msg \cup sm \times \mathcal{I} \cup \mathcal{I} \times sm) \bullet r \blacktriangle s)_\succ \\ = & \quad \{ \text{ cupjunctivity of composition; Product-split fusion 3.12a } \} \\ & (msg \bullet r \blacktriangle s \cup (sm \bullet r) \blacktriangle s \cup r \blacktriangle (sm \bullet s))_\succ \\ = & \quad \{ \ msg \bullet \mathcal{I} \times \mathcal{I} = \bot \text{; cupjunctivity of domains } \} \\ & ((sm \bullet r) \blacktriangle s)_\succ \cup (r \blacktriangle (sm \bullet s))_\succ \\ = & \quad \{ \ (r \blacktriangle s)_\succ = r_\succ \cap s_\succ \ \} \\ & ((sm \bullet r)_\succ \cap s_\succ) \cup (r_\succ \cap (sm \bullet s)_\succ) \\ = & \quad \{ \ s_\succ = \mathcal{T} \supseteq (sm \bullet r)_\succ = sp.r \ \} \\ & sp.r \cup sp.s \end{array}$$

□

Proposition 4.12 expresses that the pair of total relations $r \blacktriangle s$ can return a message on moment t whenever at least one of the constituents can return a message on that moment. Observe the join $\cup$. This is in contrast with the domain property $(r \blacktriangle s)_\succ = r_\succ \cap s_\succ$ where the meet $\cap$ emerges.
So much for the support. Next, our universe, the set of chronicles $\mathcal{C}$, will be defined. Recall that a chain $(\mathcal{S}, \preceq)$ is well-ordered if every non-empty subset of $\mathcal{S}$ has a first element, see for example Kuratowski & Mostowski [KM68]. The Fixpoint Characterisation scheme of

Theorem 3.2 is used to define $\mathcal{C}$ as the least set containing elements in $\iota \leftarrowtail \mathcal{T}$ (functions, total to ι from $\mathcal{T}$) with a well-ordered support (with respect to the order $<$), and closed under pairing with $\blacktriangle$:

Characterisation 4.13 *Set of chronicles* $\mathcal{C}$
For the base, the set of chronicles bs is defined:

$$bs \triangleq \{\, x \mid x \in \iota \leftarrowtail \mathcal{T} \ \wedge\ sp.x \text{ is well-ordered} \,\}$$

$\mathcal{C}$ is the set of chronicles such that for all predicates Pr on relations the following equivalence holds:

$$\begin{array}{ll} & \forall(\, x \ :\ x \in \mathcal{C} \ :\ Pr.x \,) \\ \equiv & \\ & \forall(\, x \ :\ x \in bs \ :\ Pr.x \,) \\ & \wedge\ \forall(\, y, z \ :\ y \in \mathcal{C} \wedge z \in \mathcal{C} \wedge Pr.y \wedge Pr.z \ :\ Pr.(y \blacktriangle z) \,) \end{array}$$

□

By convention, the identifiers f, g and h range over the chronicles in set $\mathcal{C}$. An example of a chronicle is $nm \bullet \top \bullet \mathcal{T}$; this chronicle describes *one* single channel carrying only no-messages. The function $(nm \bullet \top) \blacktriangle (nm \bullet \top) \bullet \mathcal{T}$ is a different chronicle: it describes *two* channels in parallel, both carrying nothing. Observe that the support of these two chronicles is empty: they have nothing to tell. The empty set is trivially well-ordered.
In Section 3.1.4, parallel composition was axiomatised and the pointwise definitions of the projections were given in the model. For example, the left projection $\ll$ was to be interpreted as $f \langle \ll \rangle (g, h) \equiv f = g$. Now, a problem emerges: the pair (g, h) is *not* a chronicle. It is just an element of the product $\mathcal{C} \times \mathcal{C}$. Pairing by (g, h) simply is not the way to represent pairs.
However, the set of chronicles is closed under pairing in the sense of $\blacktriangle$. This composition is an isomorphic mapping between $\mathcal{C} \times \mathcal{C}$ and the set $\{\, f \blacktriangle g \mid (f, g) \in \mathcal{C} \times \mathcal{C} \,\}$. So we do not invalidate any axioms in Chapter 3 if we switch to the notation with $\blacktriangle$. From now on, pairs in the model will be written using $\blacktriangle$. For example, the interpretation of parallel composition in the model now reads:

$$f \blacktriangle g \langle P \parallel Q \rangle\, h \blacktriangle j \ \equiv\ f \langle P \rangle\, h \wedge g \langle Q \rangle\, j$$

The condition involving $x \in bs$ results in the well-definedness of the arity of the message domain of chronicles, and in the well-orderedness of their support. These are two of the properties listed below:

Proposition 4.14
For all chronicles f:

a. $f \in \mathcal{I} \leftarrowtail \mathcal{T}$

b. $f{\prec} \subseteq \mathcal{I} \times \mathcal{I} \ \equiv\ \ll \bullet f \in \mathcal{C} \ \wedge\ \gg \bullet f \in \mathcal{C}$

c. $f{\prec} \subseteq \iota \ \equiv\ \neg(f{\prec} \subseteq \mathcal{I} \times \mathcal{I})$

d. $sp.f$ is well-ordered

Proof:
Proposition 4.14a follows from the preservation of functionality and totality by $\blacktriangle$, Theorems 3.36 and 3.41c. Propositions 4.14b and 4.14c both use the fact that the time domain $\mathcal{T}$ and the singleton message domain ι are non-empty, i.e., chronicles are non-empty. Proposition 4.14d follows from Proposition 4.12 and the fact that well-ordered sets (with respect to the *same* order) are closed under finite union. Only the proof of 4.14b is shown. Let the predicate Qr be defined by:

$$Qr.x \;\triangleq\; x{\prec} \subseteq \mathcal{I}\times\mathcal{I} \;\equiv\; {\ll}\bullet x \in \mathcal{C} \;\wedge\; {\gg}\bullet x \in \mathcal{C}$$

Then:

$$\begin{array}{ll} & \forall(\, f \;::\; Qr.f \,) \\ = & \quad \{\; \text{Characterisation 4.13} \;\} \\ & \forall(\, f \;:\; f \in bs \;:\; Qr.f \,) \wedge \forall(\, g,h \;:\; Qr.g \wedge Qr.h \;:\; Qr.(g \blacktriangle h) \,) \\ = & \quad \{\; f \in bs \text{ implies } Qr.f \;\equiv\; \bot \notin \mathcal{C}\,;\, Qr.(g\blacktriangle h) \;\equiv\; g\in\mathcal{C} \wedge h\in\mathcal{C} \;\} \\ & \forall(\, f \;:\; f \in bs \;:\; \bot \notin \mathcal{C} \,) \wedge \forall(\, g,h \;:\; Qr.g \wedge Qr.h \;:\; g\in\mathcal{C} \wedge h\in\mathcal{C} \,) \\ = & \quad \{\; \text{chronicles are non-empty; convention: } g \text{ and } h \text{ range over chronicles} \;\} \\ & \mathit{True} \end{array}$$

□

Proposition 4.14a states that all chronicles are functions, total on the time domain $\mathcal{T}$. This property will often be used in proofs. The hint in these calculation steps is 'total functions'. The second proposition 4.14b expresses that every binary-valued chronicle can be decomposed into two chronicles; or more operationally: a binary-valued channel is simply the combination of two channels side by side. The well-definedness of the arity of the left domain of a chronicle is described in 4.14c. In words it says that a chronicle is either singleton-valued or binary-valued. Together with 4.14b this ensures the uniqueness of the output arity. In Section 4.5 we will encounter a more concise description of unique arity. Finally, Proposition 4.14d describes the well-orderedness of the support of chronicles. This is equivalent to the fact that we are allowed to use induction over the support of a chronicle.
We finish the discussion on chronicles with the agreement that the set of procs $\mathcal{A}$, introduced in Chapter 3, is assumed to be equal to $\mathcal{P}(\mathcal{C}\times\mathcal{C})$ whenever calculations in the model are made. Consequently, for every interpretation $f\,\langle P\rangle\, g$ in the model it is understood that f and g are chronicles.

4.5 Axiom of Choice for Chronicles

A useful relation in $\mathcal{B}$ is the function *wipe*, which wipes a message and replaces it by a no-message of the appropriate arity. In this way, arities are maintained.

Definition 4.15 *Wipe*

$$\mathit{wipe} \triangleq \mu(\, x \;::\; nm\bullet\top\bullet\iota \cup x\times x \,)$$

□

The relation *wipe* recursively replaces all singletons by the special singleton nm. To clarify this definition, consider two of its properties:

Theorem 4.16 *Wipe*

a. $wipe \bullet \iota = nm \bullet \top \bullet \iota$

b. $wipe \bullet \mathcal{I} \times \mathcal{I} = wipe \times wipe$

□

Wiping elements twice is overdone: if all singletons in some message are replaced by nm, a second wipe will have no effect. This is expressed by the property that *wipe* is idempotent. Assumption 4.2a is equivalent to $wipe \rhd = \mathcal{I}$, which is part of the propositions for *wipe* listed next. The first proposition is the idempotency of *wipe*; the second proposition states that *wipe* is a function total on $\mathcal{I}$:

Theorem 4.17 *Wipe*

a. $wipe \bullet wipe = wipe$

b. $wipe \in \mathcal{I} \leftarrowtail \mathcal{I}$

□

Next, we rewrite Proposition 4.14c. It was motivated why this proposition expresses that chronicles have a unique output arity. Using the function *wipe*, we can express the property of unique output arity for relation r (regardless the arity of the input) as follows:

Definition 4.18 *Unique output arity*

$$\begin{array}{ll} & unar.r \\ \equiv & \\ & r \bullet \top \bullet r^{-1} \subseteq wipe^{-1} \bullet wipe \end{array}$$

□

With this definition of unique arity, Proposition 4.14c is replaced by $unar.f$ for all f:

Proposition 4.19

$$unar.f$$

□

Summarising the characteristics of chronicles we can say that chronicles are total functions (4.14a) with a well-ordered support (4.14d) and a unique output arity (4.19):

$$f \in \mathcal{I} \leftarrowtail \mathcal{T} \ \wedge \ sp.f \text{ is well-ordered } \wedge \ unar.f$$

Finally, we come to the Axiom of Choice for Chronicles. The general Axiom of Choice states that one can choose a function F out of a relation R:

$$\forall(\, R \ :: \ \exists(\, F \ : \ F \text{ is a function } : \ dom.F = dom.R \wedge F \subseteq R \,)\,)$$

To translate this general Axiom of Choice to one that is suited for choosing a chronicle f out of some relation r, we have to impose assumptions on the relation. In short, we assume that the relation r is total with a well-ordered support and a unique output arity, i.e., a 'non-deterministic chronicle'. Then, the general Axiom of Choice eliminates the non-determinism, resulting in a real chronicle. So the Axiom of Choice for Chronicles is a consequence of the general Axiom of Choice.

Assumption 4.20 *Axiom of Choice for Chronicles*
For all relations r:

$$\exists(\, f \;::\; f \subseteq r \,)$$
$$\Leftarrow$$
$$r_{\succ} = \mathcal{T} \;\wedge\; sp.r \text{ is well-ordered} \;\wedge\; unar.r$$

□

This assumption is used only sparingly; consequences of its application can be found in Sections 5.2 and 10.3.1.

Part II

Postcompose, precompose

Chapter 5

Postcompose

To be able to perform at least the normal operations on streams of messages, a 'lift' construction is needed. For example, it should be possible to lift the addition function on natural numbers $+ \in \mathbf{N} \leftarrow\!\!\!\shortmid \mathbf{N} \times \mathbf{N}$ (in $\mathcal{B}$) to a proc (in $\mathcal{A}$) that adds at every moment the two natural numbers that occur on the input g, if any, and puts the result on the output f: $f = + \bullet g$. The type of this proc should be like $I_{\mathbf{N}} \leftarrow\!\!\!\shortmid I_{\mathbf{N}} \parallel I_{\mathbf{N}}$, where $I_{\mathbf{N}}$ denotes the interface of chronicles carrying natural numbers.
The aim is that the lift operation is total on all relations in the relational algebra $\mathcal{B}$; no assumptions on the type of the argument should have to be made. In fact, one of the main reasons for the choice to denote pairs of chronicles with the split ($f \blacktriangle g$ instead of (f, g)) was the wish to give only one definition of the lift operator without conditions on the arity of the argument relation.
It turns out that the lift operator defined and investigated in this chapter is like the postcompose operation, known from (higher-order) functional calculi, extended to relations.

5.1 Postcompose defined

We use the function postcompose $\_^\circ$ to define new procs in algebra $\mathcal{A}$ from relations in algebra $\mathcal{B}$, i.e., to lift a relation.
Given some relation r, the function postcompose delivers a proc r° acting on the message domain of the input chronicle. An example of such a proc is the increment function $(1+)^\circ$ on integers. It is understood that $(1+)$ maps nm to nm; or in other words: $(1+)$ is nm-strict. The proc r° can be interpreted as follows:

$$f \,\langle r^\circ \rangle\, g \equiv \forall(\, t \;:\; t \in \mathcal{T} \;:\; f.t \,\langle r \rangle\, g.t \,)$$

Abstracting from the moments t by means of the Principle of Extensionality directly results in the definition in the model for postcompose:

Definition 5.1 *Postcompose*

$$f \,\langle r^\circ \rangle\, g \;\equiv\; f \subseteq r \bullet g$$

□

Allowing arbitrary relations in our system, especially in $\mathcal{B}$, results in generalisations of definitions covering only functional arguments. In the above case, we had to generalise the notion of postcompose known from functional calculi. The need to be able to lift arbitrary relations is apparent: some specification which does not specify a function uniquely (e.g. $\top$) can now be lifted to a proc.
The process of generalisation has its restrictions: the new notion has to be a *true* generalisation in the sense that the old notion is captured by the new one. In the specific case of Definition 5.1: if $\_^\circ$ is applied to a function, the inclusion has to be an equality. To show this, we use the relational result that *inclusion* between functions which are defined on the same source domain is in fact an *equality*:

$$\begin{array}{ll} & f \subseteq r \bullet g \\ = & \quad \{\ \mathcal{T} = f{\rhd} \subseteq (r \bullet g){\rhd} \subseteq g{\rhd} = \mathcal{T}\ \} \\ & f \subseteq r \bullet g \ \wedge\ f{\rhd} = (r \bullet g){\rhd} \\ = & \quad \{\ r \text{ is a function, so } r \bullet g \text{ is a function; above result } \} \\ & f = r \bullet g \end{array}$$

Having convinced ourselves that Definition 5.1 is right, we go on with the investigation of this function in more detail.

5.2 Basic properties of postcompose

Next, a list of the application of postcompose to the basic components of the relational algebra $\mathcal{B}$ is given. The preservation properties with respect to the product in $\mathcal{B}$ will be dealt with in the following section.

Property 5.2 *Postcompose*

a. $a^\circ = \bot\!\!\!\bot \ \equiv\ a \subseteq sm$

b. $\top^\circ = \top\!\!\!\top$

c. $(\cap(\ x\ :\ P.x\ :\ E.x\ ))^\circ = \sqcap(\ x\ :\ P.x\ :\ (E.x)^\circ\ )$

d. $(r \bullet s)^\circ \sqsupseteq r^\circ \circ s^\circ$

e. $(r \bullet s)^\circ = r^\circ \circ s^\circ \ \Leftarrow\ \ sm \bullet s \subseteq s \bullet sm \ \wedge\ s \bullet wipe^{-1} \bullet wipe \bullet s^{-1} \subseteq wipe^{-1} \bullet wipe$

f. $\mathcal{I}^\circ = I$

g. $(r^{-1})^\circ = (r^\circ)^\cup$

Proof:
Property 5.2a is a stronger result than $\bot^\circ = \bot\!\!\!\bot$. The implication $\Rightarrow$ is proved by contraposition. The proof exploits Assumption 4.2a which enables us to construct a chronicle $\in a^\circ$

carrying only no-messages. The implication $\Leftarrow$ uses the fact that supports are well-ordered but the time domain itself has no first element: every chronicle carries no-messages (of the appropriate arity) initially.
Only the statements 5.2d, 5.2e and 5.2g are proved. First, the inclusion in 5.2d is easily verified:

$$
\begin{array}{ll}
 & f \langle (r \bullet s)^\circ \rangle \, g \\
= & \quad \{ \text{ Definition 5.1 } \} \\
 & f \subseteq r \bullet s \bullet g \\
\Leftarrow & \quad \{ \text{ monotonicity } \} \\
 & \exists (\, h \;::\; f \subseteq r \bullet h \wedge h \subseteq s \bullet g \,) \\
= & \quad \{ \text{ Definition 5.1 } \} \\
 & \exists (\, h \;::\; f \langle r^\circ \rangle \, h \wedge h \langle s^\circ \rangle \, g \,) \\
= & \quad \{ \text{ interpretation composition in the model } \} \\
 & f \langle r^\circ \circ s^\circ \rangle \, g
\end{array}
$$

To prove the other implication, additional assumptions on r and s are needed. We first show 5.2g:

$$
\begin{array}{ll}
 & f \langle (r^{-1})^\circ \rangle \, g \\
= & \quad \{ \text{ Definition 5.1 } \} \\
 & f \subseteq r^{-1} \bullet g \\
= & \quad \{ \text{ Proposition 4.14a: } f, g \in \mathcal{I} \leftarrowtail \mathcal{T} \ \} \\
 & f \bullet g^{-1} \subseteq r^{-1} \\
= & \quad \{ \text{ Proposition 4.14a: } f, g \in \mathcal{I} \leftarrowtail \mathcal{T} \ \} \\
 & g^{-1} \subseteq f^{-1} \bullet r^{-1} \\
= & \quad \{ \text{ reverse } \} \\
 & g \subseteq r \bullet f \\
= & \quad \{ \text{ Definition 5.1 } \} \\
 & g \langle r^\circ \rangle \, f \\
= & \quad \{ \text{ interpretation reverse in the model } \} \\
 & f \langle (r^\circ)^\cup \rangle \, g
\end{array}
$$

The inclusion in 5.2d is due to the absence of type information about r and s. If we want to prove the inclusion $(r \bullet s)^\circ \sqsubseteq r^\circ \circ s^\circ$, we have to give some chronicle which 'fits' in between r° and s°. This is in general not possible. In 5.2e, only the assumptions on s are recorded. By Property 5.2g, the same assumptions on r^{-1} are also sufficient. The proof incorporates the Axiom of Choice for Chronicles 4.20. We assume $f \subseteq r \bullet s \bullet g$ and the conditions of 5.2e to show the existence of a chronicle h in between:

$$
\begin{array}{ll}
 & \exists (\, h \;::\; f \subseteq r \bullet h \wedge h \subseteq s \bullet g \,) \\
= & \quad \{ \text{ calculation like 5.2g } \} \\
 & \exists (\, h \;::\; h \subseteq r^{-1} \bullet f \wedge h \subseteq s \bullet g \,) \\
= & \quad \{ \text{ properties cap } \} \\
 & \exists (\, h \;::\; h \subseteq r^{-1} \bullet f \cap s \bullet g \,) \\
\Leftarrow & \quad \{ \text{ define } rr = r^{-1} \bullet f \cap s \bullet g \text{: Axiom of Choice 4.20 } \} \\
 & rr_\succ = \mathcal{T} \ \wedge \ sp.rr \text{ is well-ordered } \wedge \ unar.rr
\end{array}
$$

These three requirements are proved exploiting the assumptions. For the totality of rr:

$$
\begin{array}{ll}
 & rr\rhd \\
= & \quad \{ \text{ definition } rr \ \} \\
 & (r^{-1}\bullet f \cap s\bullet g)\rhd \\
= & \quad \{ \text{ domains: } (u\bullet v\cap w)\rhd = (v\cap u^{-1}\bullet w)\rhd \ \} \\
 & (f\cap r\bullet s\bullet g)\rhd \\
= & \quad \{ \text{ assumption } f\subseteq r\bullet s\bullet g \ \} \\
 & f\rhd \\
= & \quad \{ \text{ Proposition 4.14a: chronicles are total on } \mathcal{T} \ \} \\
 & \mathcal{T}
\end{array}
$$

The support of relation rr is well-ordered:

$$
\begin{array}{ll}
 & sp.rr \\
= & \quad \{ \ rr\rhd=\mathcal{T}\text{: Characterisation 4.11; definition } rr \ \} \\
 & (sm \ \bullet \ (r^{-1}\bullet f \cap s\bullet g))\rhd \\
\subseteq & \quad \{ \text{ monotonicity } \} \\
 & (sm\bullet s\bullet g)\rhd \\
\subseteq & \quad \{ \text{ assumption } sm\bullet s\subseteq s\bullet sm \ \} \\
 & (s\bullet sm\bullet g)\rhd \\
\subseteq & \quad \{ \text{ domains: } (u\bullet v)\rhd\subseteq v\rhd \ \} \\
 & (sm\bullet g)\rhd \\
= & \quad \{ \text{ Characterisation 4.11 } \} \\
 & sp.g
\end{array}
$$

It is a theorem that all subsets of a well-ordered set are well-ordered. Therefore, because $sp.g$ is well-ordered, it follows that $sp.rr$ is well-ordered. Finally we show that rr has a unique output arity, which is determined by g:

$$
\begin{array}{ll}
 & rr\bullet\top\bullet rr^{-1} \\
= & \quad \{ \text{ definition } rr \ \} \\
 & (r^{-1}\bullet f \cap s\bullet g) \ \bullet \ \top \ \bullet \ (r^{-1}\bullet f \cap s\bullet g)^{-1} \\
\subseteq & \quad \{ \text{ monotonicity } \} \\
 & s\bullet g\bullet\top\bullet(s\bullet g)^{-1} \\
= & \quad \{ \text{ reverse } \} \\
 & s\bullet g\bullet\top\bullet g^{-1}\bullet s^{-1} \\
\subseteq & \quad \{ \text{ Proposition 4.19 and Definition 4.18 } \} \\
 & s\bullet wipe^{-1}\bullet wipe\bullet s^{-1} \\
\subseteq & \quad \{ \text{ assumption on } s \ \} \\
 & wipe^{-1}\bullet wipe
\end{array}
$$

□

We do not have the intention to axiomatise Property 5.2b, because it can be derived from Property 5.2c, which *is* going to be axiomatised. Observe that Property 5.2c also implies that postcompose is monotonic.

The inclusion $(r \cup s)^\circ \sqsupseteq r^\circ \sqcup s^\circ$, which is also a consequence of Property 5.2c, is exactly what one expects when lifting a cup of two relations. The proc $r^\circ \sqcup s^\circ$ behaves *either* like r° *or* s°, forever. This contrasts with the behaviour of $(r \cup s)^\circ$, which can choose at every moment whether it wants to apply r or s. It might choose r always or s always, but it does not have to. In terms of predicate calculus, the inclusion in $(r \cup s)^\circ \sqsupseteq r^\circ \sqcup s^\circ$ corresponds to the implication:

$$\forall(\ y\ ::\ \exists(\ x\ ::\ P.x.y\ )\ ) \Leftarrow \exists(\ x\ ::\ \forall(\ y\ ::\ P.x.y\ )\ )$$

The assumption $s \bullet wipe^{-1} \bullet wipe \bullet s^{-1} \subseteq wipe^{-1} \bullet wipe$ in 5.2e states that the output arity of s is determined by the input arity. Because this lengthy expression occurs often, we first define the predicate $detar.r$:

Definition 5.3 *Determination of output arity*

$$\begin{array}{ll} & detar.r \\ \equiv & \\ & r \bullet wipe^{-1} \bullet wipe \bullet r^{-1} \subseteq wipe^{-1} \bullet wipe \end{array}$$

□

Several relations, such as the identities and projections, obey this determination condition:

Lemma 5.4

a. $(r \bullet a)^\circ = r^\circ \circ a^\circ \ \wedge\ (a \bullet r)^\circ = a^\circ \circ r^\circ$

b. $(r \bullet \ll)^\circ = r^\circ \circ \ll^\circ \ \wedge\ (r \bullet \gg)^\circ = r^\circ \circ \gg^\circ$

Proof:
These two statements are proved by checking the conditions of Property 5.2e. Lemma 5.4a follows from $a \subseteq \mathcal{I}$ and Corollary 3.27b. For 5.4b we verify the first condition of 5.2e:

$$\begin{array}{ll} & sm \bullet \ll \\ = & \quad \{\ \text{Product computation 3.32c}\ \} \\ & \ll \bullet\ sm \times \mathcal{I} \\ \subseteq & \quad \{\ \text{Definition 4.4}\ \} \\ & \ll \bullet\ sm \end{array}$$

The second condition, $detar.\ll$, is shown as follows:

$$\begin{array}{ll} & \ll \bullet\ wipe^{-1} \bullet wipe \bullet \ll^{-1} \\ = & \quad \{\ \text{Definition 4.15}\ \} \\ & \ll \bullet\ (nm \bullet \top \bullet \iota \cup wipe \times wipe)^{-1} \bullet\ (nm \bullet \top \bullet \iota \cup wipe \times wipe) \bullet \ll^{-1} \\ = & \quad \{\ \text{distribution and reverse; } \ll \bullet \iota = \bot\ \} \\ & \ll \bullet\ wipe^{-1} \times wipe^{-1} \bullet wipe \times wipe \bullet \ll^{-1} \\ \subseteq & \quad \{\ \text{reverse and Product computation 3.32c}\ \} \\ & wipe^{-1} \bullet \ll \bullet \ll^{-1} \bullet wipe \\ \subseteq & \quad \{\ \ll \text{ is a function}\ \} \\ & wipe^{-1} \bullet wipe \end{array}$$

□

These are examples where compositionality of postcompose holds: due to the identity or projection, there is no confusion which chronicle fits in between. In contrast, the equality $(\ll \bullet r)^\circ = \ll^\circ \circ r^\circ$ is in general *false*, for it is possible that r is not correctly typed in its second output argument.
The function postcompose lifts functions in $\mathcal{B}$ to functions in $\mathcal{A}$. We already proved some properties about postcompose in Property 5.2 which permit us not to mention any chronicle in the proof of Theorem 5.5:

Theorem 5.5 *Preservation properties*

a. $r^\circ \in a^\circ \sim b^\circ \;\Leftarrow\; r \in a \sim b$

b. r° is a function $\;\Leftarrow\; r$ is a function

c. $(r^\circ)^\rhd = (r^\rhd)^\circ \;\Leftarrow\; sm \bullet r \subseteq r \bullet sm \;\wedge\; detar.r$

Proof:
By Property 5.2f and monotonicity of postcompose we have that a° is an interface. Then:

$$
\begin{array}{ll}
 & r^\circ \in a^\circ \sim b^\circ \\
= & \quad \{ \text{ Definition 3.28 } \} \\
 & a^\circ \circ r^\circ = r^\circ \;\wedge\; r^\circ = r^\circ \circ b^\circ \\
= & \quad \{ \text{ } a \text{ and } b \text{ are identities: Lemma 5.4a } \} \\
 & (a \bullet r)^\circ = r^\circ \;\wedge\; r^\circ = (r \bullet b)^\circ \\
\Leftarrow & \quad \{ \text{ Leibniz } \} \\
 & a \bullet r = r \;\wedge\; r = r \bullet b \\
= & \quad \{ \text{ Definition 3.28 } \} \\
 & r \in a \sim b
\end{array}
$$

For 5.5b, using Definition 3.33a, we have to show that $r^\circ \circ (r^\circ)^\cup \sqsubseteq I$ follows from the inclusion $r \bullet r^{-1} \subseteq \mathcal{I}$:

$$
\begin{array}{ll}
 & r^\circ \circ (r^\circ)^\cup \\
= & \quad \{ \text{ Property 5.2g } \} \\
 & r^\circ \circ (r^{-1})^\circ \\
\sqsubseteq & \quad \{ \text{ Property 5.2d } \} \\
 & (r \bullet r^{-1})^\circ \\
\sqsubseteq & \quad \{ \text{ assumption on } r\text{; postcompose is monotonic } \} \\
 & \mathcal{I}^\circ \\
= & \quad \{ \text{ Property 5.2f } \} \\
 & I
\end{array}
$$

And for Theorem 5.5c:

$$
\begin{array}{ll}
 & (r^\circ)^\rhd \\
= & \quad \{ \text{ Definition 3.29 } \} \\
 & I \sqcap \top\!\!\top \circ r^\circ \\
= & \quad \{ \text{ Properties 5.2f and 5.2b } \}
\end{array}
$$

$$
\begin{array}{ll}
 & \mathcal{I}^\circ \sqcap \top^\circ \circ r^\circ \\
= & \quad \{ \text{ assumptions on } r\text{: Definition 5.3 and Property 5.2e } \} \\
 & \mathcal{I}^\circ \sqcap (\top \bullet r)^\circ \\
= & \quad \{ \text{ Property 5.2c } \} \\
 & (\mathcal{I} \cap \top \bullet r)^\circ \\
= & \quad \{ \text{ Definition 3.29 } \} \\
 & (r\rhd)^\circ
\end{array}
$$

Note that the inclusion $(r^\circ)\rhd \sqsubseteq (r\rhd)^\circ$ is valid without assumptions on r
□

5.3 Preservation of products

In this section the important theorem about preservation of products by postcompose is proved. This implies that the arity of the argument of postcompose is preserved, for all arities! A property, stating the preservation of the projections, is needed as a preliminary result.

Property 5.6 *Preservation of projections*

a. $\ll^\circ = \ll$

b. $\gg^\circ = \gg$

Proof:
To prove this, one is forced to take the exact definition of the projections in the model because it is not clear what the type of $\ll^\circ$ is. For all f and g:

$$
\begin{array}{ll}
 & f \langle \ll \rangle g \\
= & \quad \{ \text{ interpretation } \ll \ \} \\
 & \exists (\ h \ : \ h \in \mathcal{C} \ : \ g = f \blacktriangle h \ ) \\
= & \quad \{ \ g = f \blacktriangle h \ \Rightarrow \ f = \ll \bullet g \ \} \\
 & f = \ll \bullet g \ \wedge \ \exists (\ h \ : \ h \in \mathcal{C} \ : \ g = f \blacktriangle h \ ) \\
= & \quad \{ \text{ below: construction of chronicle } h \ \} \\
 & f = \ll \bullet g \\
= & \quad \{ \ \ll \text{ is a function } \} \\
 & f \subseteq \ll \bullet g \\
= & \quad \{ \text{ Definition 5.1 } \} \\
 & f \langle \ll^\circ \rangle g
\end{array}
$$

We postponed an implication. We show that the chronicle h we are looking for is $\gg \bullet g$. First:

$$
\begin{array}{ll}
 & f = \ll \bullet g \\
\Rightarrow & \quad \{ \text{ compose with } g^{-1}\text{: } g \text{ is a function } \} \\
 & f \bullet g^{-1} \subseteq \ll
\end{array}
$$

$$
\begin{array}{ll}
\Rightarrow & \{ \text{ domains are monotonic } \} \\
& (f \bullet g^{-1})\succ \subseteq \ll\succ \\
= & \{ \ (r \bullet s)\succ = (r\succ \bullet s)\succ;\ \ll\succ = \mathcal{I} \times \mathcal{I} \ \} \\
& (f\succ \bullet g^{-1})\succ \subseteq \mathcal{I} \times \mathcal{I} \\
= & \{ \ f \text{ and } g \text{ are total on } \mathcal{T};\ (r^{-1})\succ = r\prec \ \} \\
& g\prec \subseteq \mathcal{I} \times \mathcal{I}
\end{array}
$$

Now, the claim about $\gg \bullet g$ is verified. It follows from Proposition 4.14b that $\gg \bullet g$ is indeed a chronicle. So:

$$
\begin{array}{ll}
& f \blacktriangle (\gg \bullet g) \\
= & \{ \text{ assumption } \ f = \ll \bullet g \ \} \\
& (\ll \bullet g) \blacktriangle (\gg \bullet g) \\
= & \{ \ g \text{ is a function: Theorem 3.35c } \} \\
& \ll \blacktriangle \gg \bullet g \\
= & \{ \text{ Definition product 3.10b } \} \\
& \mathcal{I} \times \mathcal{I} \bullet g \\
= & \{ \text{ above: } g\prec \subseteq \mathcal{I} \times \mathcal{I} \ \} \\
& g
\end{array}
$$

□

The main theorem of this section follows from the preservation properties of postcompose:

Theorem 5.7 *Preservation of products*

a. $(r \blacktriangle s)^\circ = r^\circ \vartriangle s^\circ$

b. $(r \times s)^\circ = r^\circ \parallel s^\circ$

Proof:
Only the second statement is proved. A corollary of Lemma 5.4b and Property 5.6a is $(u \bullet \ll)^\circ = u^\circ \circ \ll$. Because postcompose commutes so nicely with reverse, Property 5.2g, also the equality $(\ll^{-1} \bullet u)^\circ = \ll^\cup \circ u^\circ$ is valid.

$$
\begin{array}{ll}
& (r \times s)^\circ \\
= & \{ \text{ Definition product 3.10 } \} \\
& (\ll^{-1} \bullet r \bullet \ll \cap \gg^{-1} \bullet s \bullet \gg)^\circ \\
= & \{ \text{ Property 5.2c } \} \\
& (\ll^{-1} \bullet r \bullet \ll)^\circ \sqcap (\gg^{-1} \bullet s \bullet \gg)^\circ \\
= & \{ \ (u \bullet \ll)^\circ = u^\circ \circ \ll;\ (\ll^{-1} \bullet u)^\circ = \ll^\cup \circ u^\circ \ \} \\
& \ll^\cup \circ r^\circ \circ \ll \sqcap \gg^\cup \circ s^\circ \circ \gg \\
= & \{ \text{ Definition 3.10 } \} \\
& r^\circ \parallel s^\circ
\end{array}
$$

□

It is time to go back to the initiating example of lifting the addition function on natural numbers $+ \in \mathbf{N} \leftarrow\!\!\!\shortmid \mathbf{N} \times \mathbf{N}$ (in $\mathcal{B}$) to a proc (in $\mathcal{A}$). Postcompose applied to an identity yields an interface. In the model, these interfaces can be interpreted as:

$$\forall(\, a, f \;:\; a \subseteq \mathcal{I} \;:\; f \in a^\circ \;\equiv\; f \in a \leftarrow\!\!\!\shortmid \mathcal{T}\,)$$

In particular, this holds for $\mathbf{N} \cup nm \subseteq \mathcal{I}$: $(\mathbf{N} \cup nm)^\circ$ is an interface and denotes the identity on chronicles of natural numbers (and no-messages). Now, define $I_{\mathbf{N}}$ as $(\mathbf{N} \cup nm)^\circ$, and let $+_{nm}$ be the nm-strict version of $+$. Due to the well-typed and strict function $+_{nm}$, typing of $+_{nm}$ lifts to ${+_{nm}}^\circ$ by Property 5.2e.

$$\begin{array}{ll}
 & + \in \mathbf{N} \leftarrow\!\!\!\shortmid \mathbf{N} \times \mathbf{N} \\
\Rightarrow & \quad \{ \text{ making } + \; nm\text{-strict } \} \\
 & +_{nm} \in (\mathbf{N} \cup nm) \leftarrow\!\!\!\shortmid (\mathbf{N} \cup nm) \times (\mathbf{N} \cup nm) \\
\Rightarrow & \quad \{ \text{ Theorem 5.5b; preservation of types } \} \\
 & {+_{nm}}^\circ \in (\mathbf{N} \cup nm)^\circ \leftarrow\!\!\!\shortmid ((\mathbf{N} \cup nm) \times (\mathbf{N} \cup nm))^\circ \\
= & \quad \{ \text{ Theorem 5.7b } \} \\
 & {+_{nm}}^\circ \in (\mathbf{N} \cup nm)^\circ \leftarrow\!\!\!\shortmid (\mathbf{N} \cup nm)^\circ \parallel (\mathbf{N} \cup nm)^\circ \\
= & \quad \{ \text{ definition } I_{\mathbf{N}} \; \} \\
 & {+_{nm}}^\circ \in I_{\mathbf{N}} \leftarrow\!\!\!\shortmid I_{\mathbf{N}} \parallel I_{\mathbf{N}}
\end{array}$$

So, define the addition proc by ${+_{nm}}^\circ$ and it has the desired typing properties. Notice that also other properties (laws) on $+_{nm}$ in $\mathcal{B}$, such as associativity and commutativity, carry over to ${+_{nm}}^\circ$ in $\mathcal{A}$. To get a more realistic addition proc we could first synchronise the two input chronicles (see Chapter 11) such that the addition proc ${+_{nm}}^\circ$ only gets *pairs* of naturals.

5.4 Derived notions

5.4.1 Identity on singleton channel

In Section 4.2 we defined the identity on singleton messages ι, Definition 4.1. With the postcompose operator we can lift this identity to an interface which is the identity on *singleton channels*:

Definition 5.8 *Singleton channel*

$$i \triangleq \iota^\circ$$

□

Because ι is by definition an identity, the proc i is an interface by monotonicity of postcompose. This interface i represents the set of chronicles f such that $f_{\prec} \subseteq \iota$, i.e., the base set bs in Characterisation 4.13. Therefore, it will not come as a surprise that the interface i spans the interface I:

Property 5.9 *i spans I*

$$I = \mu(\, X \;::\; i \sqcup X \parallel X\,)$$

Proof:
We convince ourselves that I is the least fixpoint of the inequation $X :: X \sqsupseteq i \sqcup X \parallel X$. Because i and $I \parallel I$ are interfaces, I solves the inequation. The proc I is characterised in the model by $f \langle I \rangle g \equiv f = g$, for all $f, g \in \mathcal{C}$. To complete the argument, it is proved that any other solution X contains the proc I:

$$
\begin{array}{ll}
 & \forall(\, f,g \;:\; f \langle I \rangle g \;:\; f \langle X \rangle g \,) \\
= & \qquad \{\; f \langle I \rangle g \equiv f = g \;\} \\
 & \forall(\, f \;::\; f \langle X \rangle f \,) \\
= & \qquad \{\; \text{Characterisation 4.13} \;\} \\
 & \forall(\, f \;:\; f \in bs \;:\; f \langle X \rangle f \,) \\
 & \quad \wedge\, \forall(\, g,h \;:\; g \langle X \rangle g \wedge h \langle X \rangle h \;:\; g \blacktriangle h \langle X \rangle g \blacktriangle h \,) \\
\Leftarrow & \qquad \{\; X \text{ solves the inequation: } i \sqsubseteq X \text{ and } X \parallel X \sqsubseteq X \;\} \\
 & \forall(\, f \;:\; f \in bs \;:\; f \langle i \rangle f \,) \\
 & \quad \wedge\, \forall(\, g,h \;:\; g \langle X \rangle g \wedge h \langle X \rangle h \;:\; g \blacktriangle h \langle X \parallel X \rangle g \blacktriangle h \,) \\
= & \qquad \{\; \text{interpretation of } i \text{ and } X \parallel X \;\} \\
 & \forall(\, f \;:\; f \in bs \;:\; f\prec \subseteq \iota \,) \\
 & \quad \wedge\, \forall(\, g,h \;:\; g \langle X \rangle g \wedge h \langle X \rangle h \;:\; g \langle X \rangle g \wedge h \langle X \rangle h \,) \\
= & \qquad \{\; \text{Characterisation 4.13: } f \in bs \Rightarrow f\prec \subseteq \iota \;\} \\
 & \mathit{True}
\end{array}
$$

This shows that I is the least solution of the inequation $X :: X \sqsupseteq i \sqcup X \parallel X$. By the Knaster-Tarski Theorem we conclude that I is also the least solution of the equation $X :: X = i \sqcup X \parallel X$
□

It is a theorem that i and $I \parallel I$ are complementary interfaces:

Theorem 5.10 *Complementary interfaces*

a. $i \sqcap I \parallel I = \perp\!\!\!\perp$

b. $i \sqcup I \parallel I = I$

Proof:

$$
\begin{array}{ll}
 & i \sqcap I \parallel I \\
= & \qquad \{\; \text{Definition 5.8; Property 5.2f} \;\} \\
 & \iota^\circ \sqcap \mathcal{I}^\circ \parallel \mathcal{I}^\circ \\
= & \qquad \{\; \text{Theorem 5.7b; Property 5.2c} \;\} \\
 & (\iota \cap \mathcal{I} \times \mathcal{I})^\circ \\
= & \qquad \{\; \text{Definition 4.1: plat calculus} \;\} \\
 & \perp^\circ \\
= & \qquad \{\; \text{Property 5.2a} \;\} \\
 & \perp\!\!\!\perp
\end{array}
$$

Theorem 5.10b is implied by Property 5.9
□

Observe that it follows from the Cone Rule that i is non-empty. Otherwise, I would have been $\perp\!\!\!\perp$ by Property 5.9. This observation also follows from Property 5.2a and the existence of no-messages. Finally, it will be shown that I is even the *unique* solution of the equation $X :: X = i \sqcup X \parallel X$:

Theorem 5.11 *Uniqueness*

$$X = I \;\equiv\; X = i \sqcup X \parallel X$$

Proof:
As stated in Theorem 5.10b, I solves the equation. Next, it is shown that there is at most one solution. Assume X and Y both solve the equation. Then it follows from Theorem 5.10a that $X \circ i = i$ and $X \circ I \parallel I = X \parallel X$ are the computation rules for X. So:

$$\begin{array}{ll}
 & X \sqsubseteq Y \\
= & \quad \{ \text{ Definition 3.4a } \} \\
 & I \sqsubseteq X \backslash Y \\
\Leftarrow & \quad \{ \text{ Property 5.9: induction } \} \\
 & i \sqcup (X\backslash Y) \parallel (X\backslash Y) \;\sqsubseteq\; X\backslash Y \\
= & \quad \{ \text{ Definition 3.4a; cupjunctivity } \} \\
 & X \circ i \;\sqcup\; X \circ (X\backslash Y) \parallel (X\backslash Y) \;\sqsubseteq\; Y \\
= & \quad \{ \text{ above-mentioned computation rules } \} \\
 & i \;\sqcup\; X \parallel X \circ (X\backslash Y) \parallel (X\backslash Y) \;\sqsubseteq\; Y \\
= & \quad \{ \text{ Parallel-parallel fusion 3.12b } \} \\
 & i \;\sqcup\; (X \circ X\backslash Y) \parallel (X \circ X\backslash Y) \;\sqsubseteq\; Y \\
\Leftarrow & \quad \{ \text{ Theorem 3.5a and monotonicity } \} \\
 & i \sqcup Y \parallel Y \;\sqsubseteq\; Y \\
= & \quad \{ \; Y \text{ solves the equation } \} \\
 & \mathit{True}
\end{array}$$

□

Observe that after having imposed Property 5.9 as an axiom, all the consequences become theorems in the calculus.

5.4.2 Equal arity

Proposition 4.14c stated that chronicles have a unique output arity. So, it is allowed to speak about *the* arity of a chronicle. Sometimes we want to know if two chronicles have the same arity, for example, if we want to merge them into a single chronicle. We define a proc *eqar* that relates any pair of chronicles with the same arity. First, two chronicles f and g both have the same arity if they represent a singleton channel:

$$\begin{array}{ll}
 & f\prec \subseteq \iota \;\wedge\; g\prec \subseteq \iota \\
= & \quad \{ \text{ calculus } \} \\
 & f \subseteq \iota \bullet \top \bullet \iota \bullet g
\end{array}$$

$$
\begin{array}{ll}
= & \{ \text{ Definition 5.1 } \} \\
& f\,\langle(\iota \bullet \top \bullet \iota)^\circ\rangle\, g \\
= & \{ \text{ } \iota \text{ is an identity: Proposition 5.2e } \} \\
& f\,\langle \iota^\circ \circ \top^\circ \circ \iota^\circ\rangle\, g \\
= & \{ \text{ Definition 5.8; } \top^\circ = \top\!\top \text{ } \} \\
& f\,\langle i \circ \top\!\top \circ i\rangle\, g
\end{array}
$$

If f and g are pairs, then their left components have to have the same arity; the same holds for their right components. This motivates the following least fixpoint definition for *eqar* that relates all chronicles of the same arity:

Definition 5.12 *Equal arity*

$$eqar \triangleq \mu(\, X \;::\; i \circ \top\!\top \circ i \sqcup X \parallel X \,)$$

□

The proc *eqar* is an equivalence relation; moreover, it 'distributes' over parallel composition:

Theorem 5.13 *Equal arity*

a. $I \sqsubseteq eqar$

b. $eqar^{\cup} = eqar$

c. $eqar \circ eqar = eqar$

d. $eqar \circ i = i \circ \top\!\top \circ i$

e. $eqar \circ I \parallel I \;=\; eqar \parallel eqar$

Proof:

$$
\begin{array}{ll}
& I \sqsubseteq eqar \\
= & \{ \text{ Property 5.9; Definition 5.12 } \} \\
& \mu(\, X \;::\; i \sqcup X \parallel X \,) \sqsubseteq \mu(\, X \;::\; i \circ \top\!\top \circ i \sqcup X \parallel X \,) \\
\Leftarrow & \{ \text{ monotonicity of the } \mu\text{-operator } \} \\
& i \sqsubseteq i \circ \top\!\top \circ i \\
= & \{ \text{ } I \sqsubseteq \top\!\top \text{ } \} \\
& \mathit{True}
\end{array}
$$

5.13b follows from Property 5.2g and the reverse-invariance of $i \circ \top\!\top \circ i$. For 5.13c we only have to show one inclusion, because the other inclusion follows from reflexivity of *eqar*, 5.13a:

$$
\begin{array}{ll}
& eqar \circ eqar \sqsubseteq eqar \\
= & \{ \text{ Definition 3.4b } \} \\
& eqar \sqsubseteq eqar / eqar
\end{array}
$$

$\Leftarrow$ { Definition 5.12: induction }
$i \circ \sqcap \circ i \sqcup (eqar/eqar) \parallel (eqar/eqar) \sqsubseteq eqar/eqar$
$=$ { Definition 3.4b; cupjunctivity }
$i \circ \sqcap \circ i \circ eqar \sqcup (eqar/eqar) \parallel (eqar/eqar) \circ eqar \sqsubseteq eqar$
$=$ { Definition 5.12 and Theorem 5.10a }
$i \circ \sqcap \circ i \circ i \circ \sqcap \circ i \sqcup (eqar/eqar) \parallel (eqar/eqar) \circ eqar \parallel eqar \sqsubseteq eqar$
$=$ { Parallel-parallel fusion 3.12b }
$i \circ \sqcap \circ i \circ i \circ \sqcap \circ i \sqcup (eqar/eqar \circ eqar) \parallel (eqar/eqar \circ eqar) \sqsubseteq eqar$
$\Leftarrow$ { Theorem 3.5b and monotonicity }
$i \circ \sqcap \circ i \sqcup eqar \parallel eqar \sqsubseteq eqar$
$=$ { Definition 5.12 }
True

Theorems 5.13d and 5.13e, which are sometimes called the computation rules, follow straightforwardly from Definition 5.12 and Theorem 5.10a
□

The proc *eqar* is the *unique* solution of the fixpoint equation $X :: X = i \circ \sqcap \circ i \sqcup X \parallel X$:

Theorem 5.14 *Uniqueness*

$$X = eqar \equiv X = i \circ \sqcap \circ i \sqcup X \parallel X$$

□

A nice corollary of the uniqueness theorem is that the proc $(wipe^{-1} \bullet wipe)^\circ$ equals the proc *eqar*, i.e., $f \langle eqar \rangle g \equiv wipe \bullet f = wipe \bullet g$ by properties of total functions:

Corollary 5.15 *Equal arity*

$$(wipe^{-1} \bullet wipe)^\circ = eqar$$

□

In the next chapter, the dual of postcompose will be defined. One of the properties of this dual, dubbed *precompose*, is that it preserves the arity of its input chronicle. In formulating this property the proc *eqar* will occur.

5.4.3 Equal support

A proc that relates chronicles with the same support is *eqsp*. It establishes a one-to-one correspondence between 'useful domains': there is an abstraction from the actual contents of the chronicles; only the moments on which messages occur are important. To define the proc *eqsp*, we first define the relation *es* by means of factors:

Definition 5.16 *Equal support*

a. $es \triangleq sm \setminus (\top \bullet sm) \cap (sm \bullet \top) / sm$

b. $eqsp \triangleq es^\circ$

□

In the model, the relation es relates any two useful messages: $a \langle es \rangle b \equiv a \in sm \equiv b \in sm$. The relation es satisfies a number of properties, among which that it is an equivalence relation:

Theorem 5.17 *es*

a. $\mathcal{I} \subseteq es$

b. $es^{-1} = es$

c. $es \bullet es \subseteq es$

d. $sm \bullet es = es \bullet sm$

e. $wipe \bullet es = es \bullet wipe$

□

The relation es is even the greatest relation that satisfies the fourth property of 5.17. The proof of the fifth statement is a bit nasty. It boils down to showing that both sides equal $wipe \bullet \top$. The interpretation of $eqsp$ reads: $f \langle eqsp \rangle g \equiv sp.f = sp.g$. The proc $eqsp$ is reflexive, symmetric and idempotent, or, in other words, it is an equivalence relation:

Theorem 5.18 *Equal support*

a. $I \sqsubseteq eqsp$

b. $eqsp^{\cup} = eqsp$

c. $eqsp \circ eqsp = eqsp$

Proof:
Reflexivity, symmetry and transitivity of $eqsp$ all follow from the corresponding properties of es, Theorem 5.17. Notice that reflexivity and transitivity imply idempotency.
□

Theorems 5.17d and 5.17e have not been exploited yet. During the discussions that follow in Chapter 10, we will encounter conditions which are implied by these two properties.

Chapter 6

Precompose

In Chapter 5, the emphasis was on the postcompose function. It was introduced to be able to perform actions on the information delivered by the input chronicle.
Another interesting feature for calculations in the model would be the ability to manipulate the *timing* of the chronicles. For example, one needs a way to describe that some output chronicle is a delayed version of some input chronicle. Let $\sigma \in \mathcal{T} \leftarrow\!\!\!\shortmid \mathcal{T}$ be such that for all $t \in \mathcal{T} \ : \ t > \sigma.t$. To express that output chronicle f is a 'σ-delayed' copy of g one could write: $f = g \bullet \sigma$, f equals g precomposed by σ.

6.1 Precompose defined

The dual of postcompose is precompose. It performs actions on the time domain, and can be used to model shifts in time:

$$f \,\langle {}^{\circ}r \rangle\, g \equiv \forall (\, t, t' \ : \ t \,\langle r \rangle\, t' \ : \ f.t' = g.t \,)$$

The above discussion suggests the following definition of precompose:

$$f \,\langle {}^{\circ}r \rangle\, g \ \equiv \ f \supseteq g \bullet r$$

However, there is a strange discontinuity in this definition of precompose when $r = \bot$. It is easy to see that the formula $f \supseteq g \bullet \bot$ is valid for all chronicles f and g because of $\bot$-strictness of composition, regardless the arities of f and g. So, it would be the case that ${}^{\circ}\bot = \top\!\top$. On the other hand, it turns out that for every non-empty relation $r \in \mathcal{T} \sim \mathcal{T}$ precompose preserves the arity:

Proposition 6.1
For $\bot \neq r \in \mathcal{T} \sim \mathcal{T}$:

$$wipe \bullet f = wipe \bullet g \ \Leftarrow \ f \supseteq g \bullet r$$

Proof:

$$\begin{array}{ll} & f \supseteq g \bullet r \\ \Rightarrow & \qquad \{ \ g \text{ is total on } \mathcal{T}\text{; typing of } r \ \} \end{array}$$

$$
\begin{array}{ll}
 & g^{-1} \bullet f \supseteq r \\
\Rightarrow & \quad \{\ r \neq \bot\text{: Cone Rule 3.19}\ \} \\
 & \top = \top \bullet g^{-1} \bullet f \bullet \top \\
\Rightarrow & \quad \{\ \text{Leibniz}\ \} \\
 & g \bullet \top \bullet f^{-1} = g \bullet \top \bullet g^{-1} \bullet f \bullet \top \bullet f^{-1} \\
\Rightarrow & \quad \{\ \text{Proposition 4.19; monotonicity}\ \} \\
 & g \bullet f^{-1} \subseteq wipe^{-1} \bullet wipe \bullet wipe^{-1} \bullet wipe \\
= & \quad \{\ f, g \text{ and } wipe \text{ are total functions: relational calculus}\ \} \\
 & wipe \bullet f = wipe \bullet g
\end{array}
$$

□

This is the discontinuity we referred to: $^\circ\bot$ relates any two chronicles, whereas $^\circ r$, for non-empty r, relates at most chronicles with the same arity. To avoid this strange behaviour, we refine our first attempt at a definition for $f \langle {}^\circ r \rangle g$ by adding the conjunct $wipe \bullet f = wipe \bullet g$:

Characterisation 6.2 *Precompose*
For $r \in \mathcal{T} \sim \mathcal{T}$:

$$f \langle {}^\circ r \rangle g \quad \equiv \quad f \supseteq g \bullet r \ \wedge \ wipe \bullet f = wipe \bullet g$$

□

Observe the type restriction on the argument of precompose: $\mathcal{T} \sim \mathcal{T}$. This restriction is not necessary, but in most of the calculations it is used. To avoid the restriction emerging in all theorems as a condition, it is encapsulated in the definition. Every time $^\circ r$ occurs, it is assumed that r is well-typed, that is, r has typing $\mathcal{T} \sim \mathcal{T}$.
Whenever it is clear that r is non-empty, we feel free to drop the second conjunct in using Characterisation 6.2 on account of Proposition 6.1.

6.2 Basic properties of precompose

Again, a list of the application of precompose to the basic components of the relational algebra $\mathcal{B}$ is given. The list is dual to Property 5.2.

Proposition 6.3

a. $^\circ\bot = eqar$

b. $^\circ(\cup(\ x\ :\ x \in S\ :\ E.x\ )) = \sqcap(\ x\ :\ x \in S\ :\ {}^\circ(E.x)\ ) \Leftarrow S \neq \emptyset$

c. $^\circ(r \cap s) \sqsupseteq {}^\circ r \sqcup {}^\circ s$

d. $^\circ(r \bullet s) \sqsupseteq {}^\circ s \circ {}^\circ r$

e. $^\circ\mathcal{T} = I$

f. $^\circ(r^{-1}) = ({}^\circ r)^\cup$

Proof:
The fourth and fifth statement are easy to verify. We will only show the other propositions. First:

$$
\begin{array}{ll}
 & f \langle {}^\circ\bot \rangle\, g \\
= & \quad \{ \text{ Characterisation 6.2 } \} \\
 & f \supseteq g \bullet \bot \ \wedge\ \mathit{wipe} \bullet f = \mathit{wipe} \bullet g \\
= & \quad \{ \ \bullet \text{ is } \bot\text{-strict } \} \\
 & \mathit{wipe} \bullet f = \mathit{wipe} \bullet g \\
= & \quad \{ \ f, g \text{ and } \mathit{wipe} \text{ are total functions: relational calculus } \} \\
 & f \subseteq \mathit{wipe}^{-1} \bullet \mathit{wipe} \bullet g \\
= & \quad \{ \text{ Definition 5.1 } \} \\
 & f \langle (\mathit{wipe}^{-1} \bullet \mathit{wipe})^\circ \rangle\, g \\
= & \quad \{ \text{ Corollary 5.15 } \} \\
 & f \langle \mathit{eqar} \rangle\, g
\end{array}
$$

Second:

$$
\begin{array}{ll}
 & f \langle {}^\circ(\cup(\, x \ :\ x \in S \ :\ E.x\,)) \rangle\, g \\
= & \quad \{ \text{ Characterisation 6.2 } \} \\
 & f \ \supseteq\ g \bullet \cup(\, x \ :\ x \in S \ :\ E.x\,) \ \wedge\ \mathit{wipe} \bullet f = \mathit{wipe} \bullet g \\
= & \quad \{ \text{ cupjunctivity of composition } \} \\
 & f \ \supseteq\ \cup(\, x \ :\ x \in S \ :\ g \bullet (E.x)\,) \ \wedge\ \mathit{wipe} \bullet f = \mathit{wipe} \bullet g \\
= & \quad \{ \text{ plat calculus } \} \\
 & \forall(\, x \ :\ x \in S \ :\ f \supseteq g \bullet (E.x)\,) \wedge \mathit{wipe} \bullet f = \mathit{wipe} \bullet g \\
= & \quad \{ \ S \text{ is non-empty: predicate calculus } \} \\
 & \forall(\, x \ :\ x \in S \ :\ f \supseteq g \bullet (E.x) \ \wedge\ \mathit{wipe} \bullet f = \mathit{wipe} \bullet g\,) \\
= & \quad \{ \text{ Characterisation 6.2 } \} \\
 & \forall(\, x \ :\ x \in S \ :\ f \langle {}^\circ(E.x) \rangle\, g\,) \\
= & \quad \{ \text{ interpretation } \sqcap \text{ in the model } \} \\
 & f \langle \sqcap(\, x \ :\ x \in S \ :\ {}^\circ(E.x)\,) \rangle\, g
\end{array}
$$

This property implies 6.3c by anti-monotonicity. For 6.3f, we use the assumption that r has typing $\mathcal{T} \sim \mathcal{T}$:

$$
\begin{array}{ll}
 & f \langle {}^\circ(r^{-1}) \rangle\, g \\
= & \quad \{ \text{ Characterisation 6.2 } \} \\
 & f \supseteq g \bullet r^{-1} \ \wedge\ \mathit{wipe} \bullet f = \mathit{wipe} \bullet g \\
= & \quad \{ \ r \in \mathcal{T} \sim \mathcal{T}\text{; reverse } \} \\
 & f \supseteq g \bullet r^{-1} \bullet \mathcal{T} \ \wedge\ \mathit{wipe} \bullet f = \mathit{wipe} \bullet g \\
= & \quad \{ \ f \text{ and } g \text{ are functions, total on } \mathcal{T} \ \} \\
 & g^{-1} \supseteq \mathcal{T} \bullet r^{-1} \bullet f^{-1} \ \wedge\ \mathit{wipe} \bullet f = \mathit{wipe} \bullet g \\
= & \quad \{ \ r \in \mathcal{T} \sim \mathcal{T}\text{; reverse; symmetry of } = \ \} \\
 & g \supseteq f \bullet r \ \wedge\ \mathit{wipe} \bullet g = \mathit{wipe} \bullet f \\
= & \quad \{ \text{ Characterisation 6.2 } \} \\
 & g \langle {}^\circ r \rangle\, f
\end{array}
$$

$$
\begin{array}{ll}
= & \{ \text{ definition reverse in the model } \} \\
& f \langle (^\circ r)^\cup \rangle\, g
\end{array}
$$

□

The relation $\mathcal{T} \bullet \top \bullet \mathcal{T}$ is the greatest relation of type $\mathcal{T} \sim \mathcal{T}$. Notice that by anti-monotonicity of precompose and Proposition 6.3e the proc $^\circ(\mathcal{T} \bullet \top \bullet \mathcal{T})$ is an interface. Interpreting it in the model we conclude that it relates *constant chronicles*:

$$
\begin{array}{ll}
& f \langle ^\circ(\mathcal{T} \bullet \top \bullet \mathcal{T}) \rangle\, f \\
= & \{ \text{ Characterisation 6.2 } \} \\
& f \supseteq f \bullet \mathcal{T} \bullet \top \bullet \mathcal{T} \\
= & \{ \ f \text{ is a total function } \} \\
& \mathcal{I} \supseteq f \bullet \top \bullet f^{-1}
\end{array}
$$

This last expression denotes that f carries the same message everywhere. Because chronicles carry at least no-messages (of some arity), that single message has to be a no-message. Therefore, the proc $^\circ(\mathcal{T} \bullet \top \bullet \mathcal{T})$ is the interface on empty chronicles. By anti-monotonicity it also follows that this interface is the *least* precompose structure. Because it is non-empty, *all* precompose structures are non-empty.
Although the preservation properties of precompose are not as nice as those of postcompose, there are some typing results. Because of the anti-monotonicity of precompose, which follows from 6.3b, and the contravariance, 6.3d, one could call precompose an anti-monotonic, contravariant relator[1], but no theory has been developed in relational algebra about such a concept. The next typing properties hold:

Proposition 6.4

a. $\quad ^\circ r \in I \leftarrow I \ \Leftarrow \ r \in \mathcal{T} \sim \mathcal{T}$

b. $\quad ^\circ r \in I \rightarrow I \ \Leftarrow \ r \in \mathcal{T} \sim \mathcal{T}$

Proof:
Assume $r \in \mathcal{T} \sim \mathcal{T}$:

$$
\begin{array}{ll}
& ^\circ r \in I \leftarrow I \\
= & \{ \text{ Definition 3.33a } \} \\
& ^\circ r \circ (^\circ r)^\cup \sqsubseteq I \\
= & \{ \text{ Proposition 6.3f } \} \\
& ^\circ r \circ {}^\circ(r^{-1}) \sqsubseteq I \\
\Leftarrow & \{ \text{ Propositions 6.3d and 6.3e } \} \\
& ^\circ(r^{-1} \bullet r) \sqsubseteq {}^\circ\mathcal{T} \\
\Leftarrow & \{ \text{ anti-monotonicity of precompose } \} \\
& r^{-1} \bullet r \supseteq \mathcal{T} \\
= & \{ \text{ relational calculus } \} \\
& r \text{ is total on } \mathcal{T}
\end{array}
$$

[1] A relator is a generalisation of the categorical notion of a functor, see Backhouse *et al.* [BdBH+91].

$\Leftarrow$ { Definition 3.38a }
$r \in \mathcal{T} \sim \mathcal{T}$

□

6.3 Preservation of arities

One of the main objectives in taking Characterisation 6.2 as definition for precompose was its preservation of arities. This is formalised in Proposition 6.5, which follows directly from Proposition 6.3a and anti-monotonicity of precompose:

Proposition 6.5

$$^\circ r \sqsubseteq eqar$$

□

Precompose preserves parallel composition in a weak sense. Probably, we ought to say that precompose-structures distribute over parallel composition. Notice that there are no restrictions (such as functionality) on the argument r:

Proposition 6.6

$$^\circ r \circ I \parallel I \;=\; {}^\circ r \parallel {}^\circ r$$

Proof:
To simplify the proof we make a case analysis. For $r = \bot$, the property follows from Proposition 6.3a and Theorem 5.13e. For $r \neq \bot$ the calculation proceeds as follows. Because $^\circ r$ preserves arities by Proposition 6.5, we know that the output chronicles of the proc $^\circ r \circ I \parallel I$ are pairs:

$$\begin{array}{ll} & f \blacktriangle g \,\langle {}^\circ r\rangle\, h \blacktriangle j \\ = & \quad \{\ r \neq \bot\text{: Characterisation 6.2}\ \} \\ & f \blacktriangle g \;\supseteq\; h \blacktriangle j \bullet r \\ = & \quad \{\ \Rightarrow\text{: Split computation 3.32; } \Leftarrow\text{: monotonicity of split}\ \} \\ & f \supseteq h \bullet r \;\wedge\; g \supseteq j \bullet r \\ = & \quad \{\ r \neq \bot\text{: Characterisation 6.2}\ \} \\ & f \,\langle {}^\circ r\rangle\, h \;\wedge\; g \,\langle {}^\circ r\rangle\, j \\ = & \quad \{\ \text{parallel composition in the model}\ \} \\ & f \blacktriangle g \,\langle {}^\circ r \parallel {}^\circ r\rangle\, h \blacktriangle j \end{array}$$

□

At first sight, this is an amazing property: even for non-deterministic relations r the equality holds, whereas one of the rules of thumb in relational calculus is that duplicating a non-deterministic relation usually increases non-determinism. However, one should bear in mind that precompose is anti-monotonic, and therefore the rule of thumb does not apply. Observe that the property would have been false, had we defined $^\circ\bot = \top\!\top$, for this would have led to the false equality $\top\!\top \circ I \parallel I \;=\; \top\!\top \parallel \top\!\top$.

It follows from Theorem 5.10b and Proposition 6.6 that the proc $^\circ r$ solves the equation $X :: X = {}^\circ r \circ i \sqcup X \parallel X$. One of the nicest results of this section is that it is even the *unique solution*:

Proposition 6.7

$$\begin{array}{ll} & X = {}^\circ r \\ \equiv & \\ & X = {}^\circ r \circ i \sqcup X \parallel X \end{array}$$

□

The proof, being similar to the proof of Theorem 5.11, is omitted. The proposition expresses that we could equally well have first defined a precompose on chronicles representing singleton channels, and, after that, closed it under parallel composition. By instantiating r to $\bot$ or $\mathcal{T}$ we get Theorems 5.11 and 5.14 once more as straightforward corollaries. This concludes the discussion on preservation of arities.

6.4 Derived notions

At the moment, the set of instruments for comparing two chronicles is very limited: I, $\ll$ and $\gg$. To be able to compare two chronicles, for example only on some interval of $\mathcal{T}$, the procs Equal-until and Equal-since are defined.

6.4.1 Preliminaries

All calculations in this subsection will be entirely on the level of the model. In the next subsection, where the main procs are defined, no calculations in the model will be needed, because all the necessary calculations will have been done in this subsection. Before continuing, we derive the following theorem, pertaining to precompose applied to an identity:

Proposition 6.8

$$^\circ(a \bullet b) = {}^\circ a \circ {}^\circ b$$

Proof:
In this proposition we encounter an example of compositionality of precompose. Let $\tilde{a}$ be $\neg a \cap \mathcal{T}$ in:

$$\begin{array}{ll} & f \langle {}^\circ a \circ {}^\circ b \rangle\, g \\ = & \quad \{ \text{ interpretation } \circ \text{ in the model } \} \\ & \exists (\, h :: f \langle {}^\circ a \rangle\, h \wedge h \langle {}^\circ b \rangle\, g \,) \\ = & \quad \{ \text{ Characterisation 6.2 } \} \\ & \exists (\, h :: f \supseteq h \bullet a \;\wedge\; \mathit{wipe} \bullet f = \mathit{wipe} \bullet h \;\wedge\; h \supseteq g \bullet b \;\wedge\; \mathit{wipe} \bullet h = \mathit{wipe} \bullet g \,) \\ = & \quad \{ \Rightarrow\text{: monotonicity; } \Leftarrow\text{: construct } h = f \bullet a \cup g \bullet b \cup \mathit{wipe} \bullet g \bullet \tilde{a} \bullet \tilde{b} \ \} \\ & f \supseteq g \bullet b \bullet a \;\wedge\; \mathit{wipe} \bullet f = \mathit{wipe} \bullet g \\ = & \quad \{ \text{ Corollary 3.27b; Characterisation 6.2 } \} \\ & f \langle {}^\circ(a \bullet b) \rangle\, g \end{array}$$

We are obliged to give some comments on the construction of h. It has to be shown that h is indeed a chronicle. The details are omitted, but we will quickly run through the required properties. First of all, h has to be total with a well-ordered support. This follows from its definition and the fact that f and g are chronicles. Secondly, h has to be a function. This follows from the assumption $f \supseteq g \bullet b \bullet a$.
□

As an example of precompose applied to an identity take some (non-empty) moment $t \subseteq \mathcal{T}$ and consider the proc $^\circ t$. In the model the proc $^\circ t$ relates every two chronicles which are equal on moment t: $f \langle ^\circ t \rangle g \equiv f \bullet t = g \bullet t$.
In the next subsection, the identities $(< \bullet t)_\prec$ and $(\geqslant \bullet t)_\prec$ will occur. These identities are complementary with respect to the time domain $\mathcal{T}$:

Proposition 6.9

a. $(< \bullet t)_\prec \cap (\geqslant \bullet t)_\prec = \bot$

b. $(< \bullet t)_\prec \cup (\geqslant \bullet t)_\prec = \mathcal{T}$

Proof:

$(< \bullet t)_\prec \cap (\geqslant \bullet t)_\prec$
= { Definition 3.29 }
$< \bullet t \bullet \top \cap \geqslant \bullet t \bullet \top \cap \mathcal{I}$
= { Proposition 4.10c: $t \bullet \top$ is a function; Theorem 3.35a }
$(< \cap \geqslant) \bullet t \bullet \top \cap \mathcal{I}$
= { Proposition 4.7a }
$\bot \bullet t \bullet \top \cap \mathcal{I}$
= { $\bot$ is zero of composition and cap }
$\bot$

And:

$(< \bullet t)_\prec \cup (\geqslant \bullet t)_\prec$
= { cupjunctivity of domains and composition }
$((< \cup \geqslant) \bullet t)_\prec$
= { Proposition 4.7b }
$(\mathcal{T} \bullet \top \bullet \mathcal{T} \bullet t)_\prec$
= { Definition 3.29 }
$\mathcal{T} \bullet \top \bullet \mathcal{T} \bullet t \bullet \top \cap \mathcal{I}$
= { $\mathcal{T} \supseteq t$ is non-empty: Cone Rule 3.19 }
$\mathcal{T} \bullet \top \cap \mathcal{I}$
= { Definition 3.29 and Theorem 3.30e }
$\mathcal{T}$

□

All these results are used to define the procs Equal-until and Equal-since.

6.4.2 Equal-until and Equal-since

The following procs in $\mathcal{A}$ are instruments to compare two chronicles with respect to a particular moment t:

Characterisation 6.10 *Equal-until, Equal-since*

a. $^{<}t \triangleq \circ((< \bullet t)^{\prec})$

b. $^{\geqslant}t \triangleq \circ((\geqslant \bullet t)^{\prec})$

□

In addition, we use the notations $^{<}r$ and $^{\geqslant}r$, meaning $^{<}(r^{\prec})$ and $^{\geqslant}(r^{\prec})$, respectively, provided that $r^{\prec}$ is a moment.
The procs $^{<}t$ and $^{\geqslant}t$ will be frequently used in calculations on the level of the model, in algebra $\mathcal{B}$. Observe that the relations $< \bullet t$ and $\geqslant \bullet t$ are non-empty for all t because of the totality of $<$ and $\geqslant$. In the model, Equal-until and Equal-since can be interpreted as:

$$f \langle {}^{<}t \rangle g \;\equiv\; \forall(\, t' \;:\; t' < t \;:\; f \bullet t' = g \bullet t' \,)$$

$$f \langle {}^{\geqslant}t \rangle g \;\equiv\; \forall(\, t' \;:\; t' \geqslant t \;:\; f \bullet t' = g \bullet t' \,)$$

The relations $(< \bullet t)^{\prec}$ and $(\geqslant \bullet t)^{\prec}$ are identities. This allows us to translate all the results of the previous subsection about precompose applied to identities to Equal-until and Equal-since. The following theorem states that $^{<}t$ and $^{\geqslant}t$ are equivalence relations (reflexive, idempotent and symmetric), and distribute over parallel composition. Furthermore, the procs $^{<}t$ and $^{\geqslant}t'$ commute: $^{<}t \circ {}^{\geqslant}t' = {}^{\geqslant}t' \circ {}^{<}t$; the special instance $^{<}t \circ {}^{\geqslant}t$ turns out to be equal to $eqar$:

Proposition 6.11
Let $\square$ be $<$ or $\geqslant$ in:

a. $I \sqsubseteq {}^{\square}t$

b. ${}^{\square}t \circ {}^{\square}t = {}^{\square}t$

c. $({}^{\square}t)^{\cup} = {}^{\square}t$

d. ${}^{\square}t \circ I \parallel I \;=\; {}^{\square}t \parallel {}^{\square}t \;=\; I \parallel I \circ {}^{\square}t$

e. ${}^{\square}t \sqsupseteq {}^{\square}t \parallel {}^{\square}t$

f. ${}^{<}t \circ {}^{\geqslant}t' = {}^{\geqslant}t' \circ {}^{<}t$

g. ${}^{<}t \circ {}^{\geqslant}t = eqar$

Proof:
We only prove the second and the last statement. The other results are direct translations of similar results for precompose, Propositions 6.3 and 6.6.

$$
\begin{array}{ll}
 & {}^{\square}t \circ {}^{\square}t \\
= & \quad \{ \text{ Characterisation 6.10 } \} \\
 & {}^{\circ}((\square \bullet t)_{\prec}) \circ {}^{\circ}((\square \bullet t)_{\prec}) \\
= & \quad \{ \ (\square \bullet t)_{\prec} \subseteq \mathcal{I} : \text{ Proposition 6.8 } \} \\
 & {}^{\circ}((\square \bullet t)_{\prec} \bullet (\square \bullet t)_{\prec}) \\
= & \quad \{ \ (\square \bullet t)_{\prec} \subseteq \mathcal{I} : \text{ Corollary 3.27a } \} \\
 & {}^{\circ}((\square \bullet t)_{\prec}) \\
= & \quad \{ \text{ Characterisation 6.10 } \} \\
 & {}^{\square}t
\end{array}
$$

And 6.11g:

$$
\begin{array}{ll}
 & {}^{<}t \circ {}^{\geqslant}t \\
= & \quad \{ \text{ Characterisation 6.10 } \} \\
 & {}^{\circ}((< \bullet t)_{\prec}) \circ {}^{\circ}((\geqslant \bullet t)_{\prec}) \\
= & \quad \{ \text{ Proposition 6.8 } \} \\
 & {}^{\circ}((< \bullet t)_{\prec} \bullet (\geqslant \bullet t)_{\prec}) \\
= & \quad \{ \text{ Corollary 3.27b and Proposition 6.9a } \} \\
 & {}^{\circ}\bot \\
= & \quad \{ \text{ Proposition 6.3a } \} \\
 & \mathit{eqar}
\end{array}
$$

□

Proposition 6.9b gives rise to a property not used in the sequel, but which might be illuminating: ${}^{<}t \sqcap {}^{\geqslant}t = I$. This means that if two chronicles are equal until some moment t and also from that moment on, then the two chronicles are equal (everywhere).
Taking the expression ${}^{<}t \circ {}^{<}t'$ and interpreting it in the model, it seems to be equivalent to ${}^{<}t \sqcup {}^{<}t'$. Equally, ${}^{\geqslant}t \circ {}^{\geqslant}t'$ seems to be equal to ${}^{\geqslant}t \sqcup {}^{\geqslant}t'$. To prove these claims without recourse to the model one lemma is used. It gives a sufficient condition for when an Equal-until structure is included in another:

Proposition 6.12

$$ {}^{<}t \sqsubseteq {}^{<}t' \ \wedge \ {}^{\geqslant}t' \sqsubseteq {}^{\geqslant}t \ \Leftarrow \ \leqslant \supseteq t' \bullet \top \bullet t $$

Proof:

$$
\begin{array}{ll}
 & {}^{<}t \sqsubseteq {}^{<}t' \ \wedge \ {}^{\geqslant}t' \sqsubseteq {}^{\geqslant}t \\
= & \quad \{ \text{ Characterisation 6.10 } \} \\
 & {}^{\circ}((< \bullet t)_{\prec}) \sqsubseteq {}^{\circ}((< \bullet t')_{\prec}) \ \wedge \ {}^{\circ}((\geqslant \bullet t')_{\prec}) \sqsubseteq {}^{\circ}((\geqslant \bullet t)_{\prec}) \\
\Leftarrow & \quad \{ \text{ anti-monotonicity of precompose } \} \\
 & (< \bullet t)_{\prec} \supseteq (< \bullet t')_{\prec} \ \wedge \ (\geqslant \bullet t')_{\prec} \supseteq (\geqslant \bullet t)_{\prec} \\
\Leftarrow & \quad \{ \text{ Definition 3.29 } \}
\end{array}
$$

$$
\begin{array}{ll}
 & < \bullet t \bullet \top \supseteq < \bullet t' \bullet \top \ \wedge \ \geqslant \bullet t' \bullet \top \supseteq \geqslant \bullet t \bullet \top \\
= & \qquad \{ \text{ Proposition 4.8; Theorem 3.6 } \} \\
 & < \bullet \leqslant \bullet t \bullet \top \bullet \top \supseteq < \bullet t' \bullet \top \ \wedge \ \geqslant \bullet \geqslant \bullet t' \bullet \top \bullet \top \supseteq \geqslant \bullet t \bullet \top \\
\Leftarrow & \qquad \{ \text{ compose with } <, \geqslant \text{ and } \top \ \} \\
 & \leqslant \bullet t \bullet \top \supseteq t' \ \wedge \ \geqslant \bullet t' \bullet \top \supseteq t \\
= & \qquad \{ \text{ Proposition 4.10c: } t \bullet \top \text{ is a total function } \} \\
 & \leqslant \supseteq t' \bullet (t \bullet \top)^{-1} \ \wedge \ \geqslant \supseteq t \bullet (t' \bullet \top)^{-1} \\
= & \qquad \{ \text{ reverse } \} \\
 & \leqslant \supseteq t' \bullet \top \bullet t
\end{array}
$$

□

In Proposition 6.12 the expression $\leqslant \supseteq t' \bullet \top \bullet t$ occurred. This is the relational expression for the more informal notation $t' \leqslant t$.

Proposition 6.13

a. $\quad {}^{<}t \circ {}^{<}t' = {}^{<}t \sqcup {}^{<}t'$

b. $\quad {}^{\geqslant}t \circ {}^{\geqslant}t' = {}^{\geqslant}t \sqcup {}^{\geqslant}t'$

c. $\quad ({}^{<}t \sqsubseteq {}^{<}t' \ \wedge \ {}^{\geqslant}t' \sqsubseteq {}^{\geqslant}t) \ \vee \ ({}^{<}t' \sqsubseteq {}^{<}t \ \wedge \ {}^{\geqslant}t \sqsubseteq {}^{\geqslant}t')$

Proof:
To prove these statements, we use properties about points and extensionality. The details will be omitted. Moreover, the next theorem from relational calculus will be used:

$$
\begin{array}{ll}
 & P \circ Q = P \sqcup Q \\
\Leftarrow & \\
 & (I \sqsubseteq P \sqsubseteq Q \ \wedge \ Q \circ Q \sqsubseteq Q) \ \vee \ (I \sqsubseteq Q \sqsubseteq P \ \wedge \ P \circ P \sqsubseteq P)
\end{array}
$$

The first two statements follow from the third statement:

$$
\begin{array}{ll}
 & {}^{<}t \circ {}^{<}t' = {}^{<}t \sqcup {}^{<}t' \ \wedge \ {}^{\geqslant}t \circ {}^{\geqslant}t' = {}^{\geqslant}t \sqcup {}^{\geqslant}t' \\
\Leftarrow & \qquad \{ \text{ above result from relational calculus } \} \\
 & ((I \sqsubseteq {}^{<}t \sqsubseteq {}^{<}t' \ \wedge \ {}^{<}t' \circ {}^{<}t' \sqsubseteq {}^{<}t') \ \vee \ (I \sqsubseteq {}^{<}t' \sqsubseteq {}^{<}t \ \wedge \ {}^{<}t \circ {}^{<}t \sqsubseteq {}^{<}t)) \\
 & \quad \wedge \ ((I \sqsubseteq {}^{\geqslant}t \sqsubseteq {}^{\geqslant}t' \ \wedge \ {}^{\geqslant}t' \circ {}^{\geqslant}t' \sqsubseteq {}^{\geqslant}t') \ \vee \ (I \sqsubseteq {}^{\geqslant}t' \sqsubseteq {}^{\geqslant}t \ \wedge \ {}^{\geqslant}t \circ {}^{\geqslant}t \sqsubseteq {}^{\geqslant}t)) \\
= & \qquad \{ \text{ Propositions 6.11a and 6.11b } \} \\
 & ({}^{<}t \sqsubseteq {}^{<}t' \vee {}^{<}t' \sqsubseteq {}^{<}t) \ \wedge \ ({}^{\geqslant}t \sqsubseteq {}^{\geqslant}t' \vee {}^{\geqslant}t' \sqsubseteq {}^{\geqslant}t) \\
\Leftarrow & \qquad \{ \text{ propositional calculus } \} \\
 & ({}^{<}t \sqsubseteq {}^{<}t' \wedge {}^{\geqslant}t' \sqsubseteq {}^{\geqslant}t) \ \vee \ ({}^{<}t' \sqsubseteq {}^{<}t \wedge {}^{\geqslant}t \sqsubseteq {}^{\geqslant}t')
\end{array}
$$

The proof of the Proposition 6.13 is completed by showing the proof of 6.13c:

$$
\begin{array}{ll}
 & ({}^{<}t \sqsubseteq {}^{<}t' \wedge {}^{\geqslant}t' \sqsubseteq {}^{\geqslant}t) \ \vee \ ({}^{<}t' \sqsubseteq {}^{<}t \wedge {}^{\geqslant}t \sqsubseteq {}^{\geqslant}t') \\
\Leftarrow & \qquad \{ \text{ Proposition 6.12 } \} \\
 & \leqslant \supseteq t' \bullet \top \bullet t \ \vee \ \leqslant \supseteq t \bullet \top \bullet t' \\
= & \qquad \{ \text{ reverse } \} \\
 & \leqslant \supseteq t' \bullet \top \bullet t \ \vee \ \geqslant \supseteq t' \bullet \top \bullet t
\end{array}
$$

$$
\begin{array}{ll}
= & \{\ t \bullet \top \text{ is a point: Principle of Extensionality } \} \\
& \leqslant \cup \geqslant \supseteq t' \bullet \top \bullet t \\
\Leftarrow & \{\ \text{Proposition 4.10a: monotonicity } \} \\
& \leqslant \cup \geqslant \supseteq \mathcal{T} \bullet \top \bullet \mathcal{T} \\
= & \{\ \text{Proposition 4.7b } \} \\
& \mathit{True}
\end{array}
$$

□

According to Proposition 6.11a, all ${}^{<}t$ structures include I. Taking an arbitrary cap over all t one expects to get I. Dually, all ${}^{<}t$ structures are included in *eqar*, Proposition 6.5. Bringing all those ${}^{<}t$ in one cup, we expect to get *eqar*, taking into account that chronicles have a well-ordered support:

Proposition 6.14

a. $I = \sqcap(\, t \ ::\ {}^{<}t \,)$

b. $\mathit{eqar} = \sqcup(\, t \ ::\ {}^{<}t \,)$

Proof:

$$
\begin{array}{ll}
& \sqcap(\, t \ ::\ {}^{<}t \,) \\
= & \{\ \text{Characterisation 6.10a; } t \in \mathcal{T} \neq \bot\text{: Proposition 6.3b } \} \\
& {}^{\circ}(\cup(\, t \ ::\ (<\bullet t)_{\prec} \,)) \\
= & \{\ \text{cupjunctivity of domains and composition } \} \\
& {}^{\circ}((< \bullet \cup(\, t \ ::\ t \,))_{\prec}) \\
= & \{\ \text{Extensionality } \} \\
& {}^{\circ}((<\bullet \mathcal{T})_{\prec}) \\
= & \{\ \text{Assumption 4.5: } < \text{ is total and surjective on } \mathcal{T} \ \} \\
& {}^{\circ}\mathcal{T} \\
= & \{\ \text{Proposition 6.3e } \} \\
& I
\end{array}
$$

To prove the second statement we need a result expressing the fact that every chronicle has an *initial* section with no-messages. This follows from the assumption that non-empty well-ordered sets have a first element:

$$
\begin{array}{ll}
& \sqcup(\, t \ ::\ {}^{<}t \,) = \mathit{eqar} \\
= & \{\ \text{Characterisation 6.10a; Proposition 6.5 } \} \\
& \sqcup(\, t \ ::\ {}^{<}t \,) \sqsupseteq \mathit{eqar} \\
\Leftarrow & \{\ \text{Theorems 5.14 and 3.2; monotonicity } \} \\
& \sqcup(\, t \ ::\ {}^{<}t \,) \ \sqsupseteq\ i \circ \top\!\!\top \circ i \sqcup \sqcup(\, t \ ::\ {}^{<}t \,) \parallel \sqcup(\, t \ ::\ {}^{<}t \,) \\
= & \{\ \text{properties } \sqcup \ \} \\
& \sqcup(\, t \ ::\ {}^{<}t \,) \sqsupseteq i \circ \top\!\!\top \circ i \ \wedge\ \sqcup(\, t \ ::\ {}^{<}t \,) \sqsupseteq \sqcup(\, t \ ::\ {}^{<}t \,) \parallel \sqcup(\, t \ ::\ {}^{<}t \,) \\
= & \{\ \text{Proposition 6.13c: diagonalisation } \} \\
& \sqcup(\, t \ ::\ {}^{<}t \,) \sqsupseteq i \circ \top\!\!\top \circ i \ \wedge\ \sqcup(\, t \ ::\ {}^{<}t \,) \sqsupseteq \sqcup(\, t \ ::\ {}^{<}t \parallel {}^{<}t \,) \\
= & \{\ \text{Proposition 6.11e } \}
\end{array}
$$

$$
\begin{array}{ll}
 & \sqcup(\, t \;::\; {}^{<}t \,) \sqsupseteq i \circ \sqcap\!\!\sqcap \circ i \\
= & \qquad \{ \text{ interpretation in the model: for all } f \text{ and } g \ \} \\
 & \exists(\, t \;::\; f \bullet (< \bullet\, t)^{\prec} = g \bullet (< \bullet\, t)^{\prec} \,) \;\Leftarrow\; f, g \in bs
\end{array}
$$

According to Characterisation 4.13, f and g have a well-ordered support. If both supports are empty, f and g are equal (to $nm \bullet \top \bullet \mathcal{T}$). If at least one of the supports is non-empty, it suffices to take the minimum of the support of f and g. Below that minimum, f and g both carry only no-messages nm, and so they are equal

□

So much for some of the basic elements of the calculus. We presented useful functions to build procs: postcompose and precompose. Furthermore, several procs to compare chronicles were introduced: $eqar$, $eqsp$, ${}^{<}t$ and ${}^{\geqslant}t$. In the next part, some healthiness aspects of the calculus are discussed. Then, it will be possible to tackle the preservation problems of feedback.

Summarising the results

We summarise the derived propositions of the function postcompose. Remember that the function precompose acts on the time domain, and, therefore, is not part of the calculus.

Axiom 6.15

a. $a^\circ = \perp\!\!\!\perp \ \equiv \ a \subseteq sm$

b. $(\cap(\ x \ : \ P.x \ : \ E.x\ ))^\circ = \sqcap(\ x \ : \ P.x \ : \ (E.x)^\circ\ )$

c. $(r \bullet s)^\circ \sqsupseteq r^\circ \circ s^\circ$

d. $(r \bullet s)^\circ = r^\circ \circ s^\circ \ \Leftarrow \ \ sm \bullet s \subseteq s \bullet sm \ \wedge \ detar.s$

e. $\mathcal{I}^\circ = I$

f. $(r^{-1})^\circ = (r^\circ)^\cup$

g. $\ll^\circ = \ll$

h. $\gg^\circ = \gg$

i. $I = \mu(\ X \ :: \ i \sqcup X \parallel X\ )$

□

All the other properties in this part on postcompose can be derived from this set of axioms.

Part III

Healthiness

Chapter 7

Causality

In Section 3.5 two problems were encountered: functionality and totality are not preserved by feedback. In this chapter the notion of *causality* is introduced. This condition on procs should establish the following properties:

Firstly, feedback should preserve functionality and totality:

$$F^\sigma \text{ is a function } \Leftarrow \mathit{causal}.F$$

$$P^\sigma \in A \rightsquigarrow B \;\Leftarrow\; \mathit{causal}.P \;\wedge\; P \in A \rightsquigarrow B \parallel A$$

Secondly, causality itself should be preserved by all the composition operations, in particular by the feedback construction:

$$\mathit{causal}.(P^\sigma) \Leftarrow \mathit{causal}.P$$

Finally, the notion of causality should not be unnecessarily restricted.
It will turn out that the properties listed above are too strong. We need stronger (typing) conditions. Furthermore, the consequence of, in particular, the functionality of feedback has to be weakened. This all is due to the inherent *polymorphism* of the system.
The discussion on causality also appeared in Rietman [Rie93a]. In that technical report, the model differed significantly from the one used in this thesis: it was assumed that there is a minimal moment t_0 in the time domain. In Chapter 4, this assumption has been replaced by the well-orderedness of the *support*.

7.1 Pinpointing the exact problems

Feedback does not preserve functionality; recall the counterexample of the feedback of the function $\gg$ in Section 3.5: $\gg^\sigma = \top\top$. Apparently, our notion of functionality does not match with the notion of 'implementability'. And indeed, when we take a closer look at the projections, we find several unrealistic properties.
One problem with the projections is that they react instantaneously to the input. This is not a reasonable property of any implementable process: every physical machine has some *delay*. This delay will be modelled by *archimedean functions*.

But having some delay is not enough to preserve functionality. Polymorphism is also the cause of some problems. Consider the function $^\circ(-1)\circ\gg$, which takes its second input channel and delays it for one time unit before sending it to the output. Placing this delaying function in a feedback loop results in a non-deterministic proc: $(^\circ(-1)\circ\gg)^\sigma$ equals $(^\circ(-1)\sqcap I)\circ\top\!\!\top$. The interface $^\circ(-1)\sqcap I$ describes the set of empty chronicles. This follows from the equality $f = f\bullet(-1)$ and the fact that the support of chronicles is well-ordered. Therefore, the proc $(^\circ(-1)\sqcap I)\circ\top\!\!\top$ is non-deterministic in the sense that it is able to return an empty chronicle of arbitrary output arity.
At this point there are two possibilities. We can require functions to have a unique output arity; this cancels out the output polymorphism. Or we could weaken our notion of functionality: observe that the procs $^\circ(-1)\circ\gg$ and $(^\circ(-1)\sqcap I)\circ\top\!\!\top$ are both *functional per output arity*. Because we prefer to calculate with polymorphism as long as possible, we take the second possibility. The property of functionality per output arity is dubbed *polyfunctionality*. It is the case that the feedback of a delaying polyfunction is again a polyfunction.

Feedback does not preserve totality. Again, polymorphism is the main problem. Consider the total delaying function $I\vartriangle I\circ{}^\circ(-1)\circ\gg$, which takes its second input channel and delays it for one time unit before sending it *as a pair* to the output. Placing this delaying total function in a feedback loop results in the non-total function $\perp\!\!\perp$: $(I\vartriangle I\circ{}^\circ(-1)\circ\gg)^\sigma = \perp\!\!\perp$. The problem is that the arity is 'doubled' every iteration of the loop, but it is forced by the feedback to stay the same. The only candidates for the arity of the output are therefore the zero arity and an infinite arity. Both zero and infinite arities are impossible by Characterisation 4.13 and therefore the feedback equals $\perp\!\!\perp$. The solution is to require the second input arity and the output arity to be the same. This typing information will be provided by *primed typing*.
Having an agreement on input arity and output arity is still insufficient to get preservation of totality. Because the feedback builds its output in an iteration loop, we need the existence of the limit of these successive iterations. Consider the proc *One* which is defined on *finite* chronicles of 1's. The proc copies its input with some delay to the output, but also produces an additional message 1 once. When the proc $One\circ\gg$ is placed in a feedback loop, the output of the loop can only be a continuous stream of 1's. But then, the proc *One* has to be defined on this limit of finite chronicles of 1's. A proc which is also defined on the limits of its finite input chronicles is called *closed*.

The following Sections 7.2 and 7.3 will discuss archimedean functions to model the delay of a proc, and primed typing. Finally, the notion of causality is explained, defined and investigated.

7.2 Archimedean functions

In the previous section we showed that an instantaneously reacting proc can cause problems, and that it is reasonable to assume some delay. To capture the notion of *progress*

in time of a proc archimedean functions are introduced. First, the notion of an *order-isomorphism* is defined:

Characterisation 7.1 *Order-isomorphism*

$$\begin{array}{l} \quad oiso.\sigma \\ \equiv \\ \quad \sigma \bullet \sigma^{-1} = \mathcal{T} = \sigma^{-1} \bullet \sigma \ \wedge \ \sigma \bullet < = < \bullet \sigma \end{array}$$

□

The first conjunct states that σ is an isomorphism on $\mathcal{T}$: it is total and surjective, functional and injective. The second conjunct describes monotonicity of σ with respect to the order $<$ and is interpreted as follows: for all $t, t' \in \mathcal{T}$,

$$t < t' \equiv \sigma.t < \sigma.t'$$

Among the most important propositions of order-isomorphisms is their closedness under sequential composition and reverse:

Proposition 7.2

Order-isomorphisms are closed under sequential composition and reverse

□

Now, the following definition for archimedeans is taken:

Characterisation 7.3 *Archimedean function*

$$\begin{array}{l} \quad arch.\alpha \\ \equiv \\ \quad \alpha \subseteq > \ \wedge \ oiso.\alpha \end{array}$$

□

Identifiers α and β range over archimedeans. It is assumed that archimedean functions do exist in $\mathcal{B}$. The conjunct $\alpha \subseteq >$ can be interpreted as:

$$\forall(\, t \ :: \ \alpha.t > t \,)$$

This expresses the progress. All the assumed properties of archimedean functions have many interesting implications. For example, it is not possible that a sequence $(\alpha^n \bullet t)_n$ has accumulation points. Starting from some moment t, every moment t' can be exceeded by a finite number of iterations of α:

$$\forall(\, t, t' \ :: \ \exists(\, n \ : \ n \in \mathbf{N} \ : \ t' < \alpha^n.t \,) \,)$$

This is a consequence of the assumption that the time domain $\mathcal{T}$ is closed, Assumption 4.5. In terms of the algebra, the unboundedness of α reads:

Proposition 7.4

$$\forall(\, \alpha \ :: \ < \bullet \alpha^* = \mathcal{T} \bullet \top \bullet \mathcal{T} \,)$$

□

From the progress property we conclude that whenever two chronicles are related by ${}^<(\alpha \bullet t)$ they also have to be related by ${}^<t$, for all t. A generalisation of this property reads:

Proposition 7.5

$$ {}^<(\sigma \bullet t) \sqsubseteq {}^<t \;\Leftarrow\; \sigma \in \mathcal{T} \leftrightarrow\!\!\!\!\!\!\rightarrow \mathcal{T} \;\wedge\; \sigma \subseteq \geqslant $$

Proof:
From the fact that σ is a total function we conclude that $(\sigma \bullet t)_\prec$ is again a moment (interpreted as $\sigma.t$). Actually, a proof is required, but we take that for granted from relational calculus. It follows that ${}^<(\sigma \bullet t)$ is a permissible expression.

$$
\begin{array}{ll}
 & {}^<(\sigma \bullet t) \sqsubseteq {}^<t \\
\Leftarrow & \quad \{ \text{ Proposition 6.12 } \} \\
 & \geqslant \supseteq t \bullet \top \bullet (\sigma \bullet t)_\prec \\
= & \quad \{ \text{ reverse; } s_\prec \bullet \top = s \bullet \top \} \\
 & \geqslant \supseteq \sigma \bullet t \bullet \top \bullet t \\
= & \quad \{ \text{ Proposition 4.10b } \} \\
 & \geqslant \supseteq \sigma \bullet t \\
\Leftarrow & \quad \{ \text{ } t \text{ is an identity } \} \\
 & \geqslant \supseteq \sigma
\end{array}
$$

□

In Proposition 6.11a the property $I \sqsubseteq {}^<t$ was derived. This implies for all $n \in \mathbf{N}$ and for all t: $I \sqsubseteq {}^<(\alpha^n \bullet t)$. Or equivalently: $I \sqsubseteq \sqcap(\, n \;:\; n \in \mathbf{N} \;:\; {}^<(\alpha^n \bullet t)\,)$ for all t. As may be expected, the cap-structure actually *equals* the identity, keeping in mind the unboundedness of α:

Proposition 7.6

$$ I = \sqcap(\, n \;:\; n \in \mathbf{N} \;:\; {}^<(\alpha^n \bullet t)\,) $$

Proof:

$$
\begin{array}{ll}
 & \sqcap(\, n \;:\; n \in \mathbf{N} \;:\; {}^<(\alpha^n \bullet t)\,) \\
= & \quad \{ \text{ Characterisation 6.10a; } \mathbf{N} \neq \emptyset \text{: Proposition 6.3b } \} \\
 & {}^\circ(\cup(\, n \;:\; n \in \mathbf{N} \;:\; (< \bullet \alpha^n \bullet t)_\prec\,)) \\
= & \quad \{ \text{ cupjunctivity of domains and composition } \} \\
 & {}^\circ((< \bullet \cup(\, n \;:\; n \in \mathbf{N} \;:\; \alpha^n\,) \bullet t)_\prec) \\
= & \quad \{ \text{ Definition 3.7c } \} \\
 & {}^\circ((< \bullet \alpha^* \bullet t)_\prec) \\
= & \quad \{ \text{ Proposition 7.4 } \} \\
 & {}^\circ((\mathcal{T} \bullet \top \bullet \mathcal{T} \bullet t)_\prec) \\
= & \quad \{ \text{ Definition 3.29 } \} \\
 & {}^\circ(\mathcal{T} \bullet \top \bullet \mathcal{T} \bullet t \bullet \top \cap \mathcal{I}) \\
= & \quad \{ \text{ } \mathcal{T} \supseteq t \text{ is non-empty: Cone Rule 3.19 } \} \\
 & {}^\circ(\mathcal{T} \bullet \top \cap \mathcal{I}) \\
= & \quad \{ \text{ Definition 3.29 and Theorem 3.30e } \}
\end{array}
$$

$$
\begin{array}{ll}
 & {}^{\circ}\mathcal{T} \\
= & \quad \{ \text{ Proposition 6.3e } \} \\
 & I
\end{array}
$$

□

Compare this result and its proof to Proposition 6.14a. This concludes the discussion about archimedeans.

7.3 Primed typing

In functional programming languages, the type (information) of a function is one of the fundamental concepts. In particular, the notion of *polymorphism* has an prominent role, see for example Bird & Wadler [BW88]. The identity function *id* has the polymorphic type $\alpha \to \alpha$, where α is a so-called type variable. A type variable can be instantiated to different types in different circumstances: the expression $id.3$ is well-defined and has type $\mathbf{N}$ because $\mathbf{N}$ can be *substituted* for α in the polymorphic type of *id*. In general, for an input element of type A one gets as output something of type A, for any A.
In the system that we advocate, there is no such thing as the polymorphic type of a relation: the identity I is *one* function which is the identity on *every* element (of the universe $\mathcal{C}$). However, it could be useful to formalise in a type expression 'for an input element of type A one gets as output something of type A' in case of the identity I. In particular, we are interested in the *arity preserving* properties of I: 'for input element with some arity one gets as output something with the same arity'.
We are heading for a new notion of typing dubbed *primed typing* which gives us the desired arity preservation information: for some proc P, the typing $P \in A' \sim B'$ should imply firstly that P has type $A \sim B$, and secondly, that P preserves the arity of its input element (it might be defined on several arities). In a relational expression this reads $P \in A \sim B \ \wedge \ P \sqsubseteq eqar$. With this meaning, it follows from Theorem 5.13a that I has the typing $I' \sim I'$.
In the rest of this section we will define precisely what the meaning is of primed typing. First of all, the set $\mathcal{D}$ is defined by a fixpoint characterisation, using the scheme of Theorem 3.2. It contains all the greatest, well-typed interfaces; well-typed in the sense of well-defined arity:

Definition 7.7 *Well-typed interfaces*
For all predicates Pr on procs, the following equivalence holds:

$$
\begin{array}{ll}
 & \forall(\, X \ : \ X \in \mathcal{D} \ : \ Pr.X \,) \\
\equiv & \\
 & Pr.(i) \wedge \forall(\, X,Y \ : \ X \in \mathcal{D} \wedge Y \in \mathcal{D} \wedge Pr.X \wedge Pr.Y \ : \ Pr.(X \parallel Y) \,)
\end{array}
$$

□

For example, the interfaces i and $(i \parallel i) \parallel i$ are in $\mathcal{D}$; the interfaces $\perp\!\!\!\perp$ and I are not elements of $\mathcal{D}$ because there does not seem to be a reasonable arity for them. Taking arbitrary cups

in several ways gives us known procs:

Lemma 7.8

a. $\sqcup(\, X \;:\; X \in \mathcal{D} \;:\; X \,) = I$

b. $\sqcup(\, X \;:\; X \in \mathcal{D} \;:\; X \circ \top\!\!\top \circ X \,) = \mathit{eqar}$

Proof:
Theorems 5.11 and 5.14 and Definition 7.7 imply the desired results
□

Other results such as $\sqcup(\, X \;:\; X \in \mathcal{D} \;:\; X \circ \top\!\!\top \circ I \parallel X \,) = \mathit{eqar} \circ \gg$ follow from this lemma by cupjunctivity properties of parallel composition and sequential composition.
The set $\mathcal{D}$ and Lemma 7.8 open the way for defining what we mean by primed typing.
It is assumed that "$A^{(n)}$" stands for an expression (syntax) of an ordinary interface A decorated with n primes, for $1 \leqslant n$. We let identifiers Φ and Ψ range over the expressions of (possibly) primed interfaces. Substitutions are used to manipulate decorated expressions. For $X \in \mathcal{D}$, the substitution $[X/v^{(n)}]$, for $1 \leqslant n$, matches any n-primed interface in an expression and replaces it by X. There is an implicit quantification of the variable v which matches interfaces. The substitution $[v/v^{(i)}]$, which also has an implicit quantification of i, removes all primes in the expression to which it is applied.
For example, take the expression "$A' \circ \top\!\!\top \circ B'' \parallel C'$", which is actually a piece of syntax. Here, A, B and C denote ordinary interfaces, Definition 3.25, which are decorated with a number of primes. Applying substitution $[X/v^{(1)}]$ results in the expression "$X \circ \top\!\!\top \circ B'' \parallel X$"; afterwards, the substitution $[v/v^{(i)}]$ removes the remaining primes, and the result is interpreted as the *proc* $X \circ \top\!\!\top \circ B \parallel X$.

Definition 7.9 *Primed typing*

$$P \in \Phi \sim \Psi$$
$$\equiv$$
$$P \sqsubseteq \sqcap(\, n \;:\; 1 \leqslant n \;:\; \mathit{Sb}.n.(\Phi, \Psi) \,)$$

where the function *substitute* Sb is defined for all n, Φ and Ψ by:

$$\mathit{Sb}.n.(\Phi, \Psi) = \sqcup(\, X \;:\; X \in \mathcal{D} \;:\; (\Phi \circ \top\!\!\top \circ \Psi)[X/v^{(n)}][v/v^{(i)}] \,)$$

□

For example, $\mathit{Sb}.1.(A', B'' \parallel C')$ results in the proc $\sqcup(\, X \;:\; X \in \mathcal{D} \;:\; X \circ \top\!\!\top \circ B \parallel X \,)$, which, by Lemma 7.8, can be transformed to $\mathit{eqar} \circ \gg \circ B \parallel I$, while $\mathit{Sb}.2.(A', B'' \parallel C')$ results in $\sqcup(\, X \;:\; X \in \mathcal{D} \;:\; A \circ \top\!\!\top \circ X \parallel C \,)$.
The reader is urged to check that primed typing, Definition 7.9, is a real extension of ordinary typing, Definition 3.28. To clarify the new definition and the role of function Sb, we present a few results needed in the next section on causality:

Lemma 7.10

a. $P \in A' \sim B' \;\equiv\; P \sqsubseteq A \circ \mathit{eqar} \circ B$

b. $P \in A' \sim B \parallel C' \;\equiv\; P \sqsubseteq A \circ \mathit{eqar} \circ \gg \circ B \parallel C$

Proof:
We only show 7.10a:

$$
\begin{array}{ll}
 & P \in A' \sim B' \\
= & \quad \{ \text{ Definition 7.9 } \} \\
 & P \sqsubseteq \sqcap(\, n \ : \ 1 \leqslant n \ : \ Sb.n.(A', B') \,) \\
= & \quad \{ \text{ plat calculus } \} \\
 & P \sqsubseteq Sb.1.(A', B') \sqcap \sqcap(\, n \ : \ 2 \leqslant n \ : \ Sb.n.(A', B') \,) \\
= & \quad \{ \text{ definition } Sb \text{ in Definition 7.9; } v^{(1)} = v' \ \} \\
 & P \sqsubseteq \sqcup(\, X \ : \ X \in \mathcal{D} \ : \ (A' \circ \top\!\!\top \circ B')[X/v'][v/v^{(i)}] \,) \\
 & \quad \sqcap \sqcap(\, n \ : \ 2 \leqslant n \ : \ \sqcup(\, X \ : \ X \in \mathcal{D} \ : \ (A' \circ \top\!\!\top \circ B')[X/v^{(n)}][v/v^{(i)}] \,) \,) \\
= & \quad \{ \text{ substitutions } [X/v'] \text{ and } [X/v^{(n)}] \text{ for } 2 \leqslant n \ \} \\
 & P \sqsubseteq \sqcup(\, X \ : \ X \in \mathcal{D} \ : \ (X \circ \top\!\!\top \circ X)[v/v^{(i)}] \,) \\
 & \quad \sqcap \sqcap(\, n \ : \ 2 \leqslant n \ : \ \sqcup(\, X \ : \ X \in \mathcal{D} \ : \ (A' \circ \top\!\!\top \circ B')[v/v^{(i)}] \,) \,) \\
= & \quad \{ \text{ substitution } [v/v^{(i)}] \ \} \\
 & P \sqsubseteq \sqcup(\, X \ : \ X \in \mathcal{D} \ : \ X \circ \top\!\!\top \circ X \,) \\
 & \quad \sqcap \sqcap(\, n \ : \ 2 \leqslant n \ : \ \sqcup(\, X \ : \ X \in \mathcal{D} \ : \ A \circ \top\!\!\top \circ B \,) \,) \\
= & \quad \{ \text{ Lemma 7.8b; plat calculus } \} \\
 & P \sqsubseteq \mathit{eqar} \sqcap A \circ \top\!\!\top \circ B \\
= & \quad \{ \text{ Theorem 3.26b: } Q \sqcap A \circ \top\!\!\top \circ B \ = \ A \circ Q \circ B \ \} \\
 & P \sqsubseteq A \circ \mathit{eqar} \circ B
\end{array}
$$

□

Several preservation rules for primed typing such as:

$$
\begin{array}{ll}
 & P \circ Q \ \in \ I' \sim I' \parallel I \\
\Leftarrow & \\
 & P \ \in \ I' \sim I' \parallel I \ \ \wedge \ \ Q \in I' \sim I'
\end{array}
$$

are valid. This result can be obtained by applying Definition 7.9 directly, but a better way would be to give some *unification algorithm* which makes it possible to reason at the level of the types only.
A corollary is the primed typing of I: because $I \sqsubseteq I \circ \mathit{eqar} \circ I$ is valid, Theorem 5.13a, we can deduce from 7.10a that the typing $I \in I' \sim I'$ is correct, just as we wanted. The rhs of Lemma 7.10b implies that the second input arity and the output arity of proc P are the same. It was explained in Section 7.1 that this information is needed for reasoning about P^{σ}.

7.4 Causality for polyfunctions

In Section 7.1 it was shown that plain functionality is not preserved by feedback. The property of being functional per output arity was more appropriate. Functionality per

output arity is described by polyfunctionality:

Definition 7.11 *Polyfunction*

$$P \text{ is a polyfunction}$$
$$\equiv$$
$$eqar \sqcap P \circ P^{\cup} \sqsubseteq I$$

□

There are several ways to deduce functionality from polyfunctionality. The next theorem states a sufficient assumption:

Theorem 7.12 *Polyfunction*

$$P \text{ is a function} \quad \equiv \quad P \circ P^{\cup} \sqsubseteq eqar \ \wedge \ P \text{ is a polyfunction}$$

□

Without loss of reasonability, the assumption is made that the present output of implementable processes is determined and only depends on past input. This property is described by *inertia* of P: there exists some α such that for all moments t and for any two possible input chronicles f and g which are equal until moment t, any two possible output chronicles of P ('$P.f$' and '$P.g$') with the same arity are equal until a *later* (because $\alpha \subseteq >$) moment $\alpha.t$. So, P does not react instantaneously, nor does its output depend on future input:

Characterisation 7.13 *Inert*

$$inert.P$$
$$\equiv$$
$$\exists(\ \alpha \ ::\ \forall(\ t \ ::\ eqar \sqcap P \circ {<}t \circ P^{\cup} \sqsubseteq {<}(\alpha \bullet t)\ )\ )$$

□

Several alternative formulations for inertia have been studied. Consider the following three equivalent formulations for some archimedean function α:

$$\forall(\ t \ ::\ P \circ {<}t \circ P^{\cup} \sqsubseteq {<}(\alpha \bullet t)\ )$$

$$\forall(\ t \ ::\ P \circ {<}t \circ P{>} \sqsubseteq {<}(\alpha \bullet t) \circ P\ ) \ \wedge \ P \text{ is a function}$$

$$\forall(\ t \ ::\ eqar \sqcap P \circ {<}t \circ P^{\cup} \sqsubseteq {<}(\alpha \bullet t)\ ) \ \wedge \ \text{input arity determines output arity of } P$$

The interpretation of the first alternative reads: for all moments t and for any two possible input chronicles f and g which are equal until moment t, any two possible output chronicles of P are equal until a later moment $\alpha.t$. This alternative is wrong: it is not preserved by feedback. A counterexample is the (total) function $^{\circ}(-1) \circ \gg$: $(^{\circ}(-1) \circ \gg)^{\sigma}$ is not a function, and, therefore, it can not be inert according to the first alternative.
We could weaken the first alternative by dropping functionality: take the first conjunct of the second formulation. This alternative is also wrong: preservation of totality by the

feedback can not be guaranteed. A counterexample is the total proc $M \circ \gg$, where M ('*M*onster') is defined by $f \langle M \rangle g \equiv$ (f finishes $\equiv$ g does not finish): $(M \circ \gg)^\sigma = \perp\!\!\!\perp$.
Another possibility is weakening the first alternative by dropping determination of arities. This results in Characterisation 7.13. It corresponds to *prefix monotonicity* used by, for example, Broy [Bro90]: if we have observed a finite (up to some moment $\alpha.t$) output chronicle for a corresponding finite input chronicle (up to some moment t), then if we observe additional input (up to a moment $t' > t$) we may just observe additional output (up to a moment $\geqslant \alpha.t$).
Notice that the property of inertia is a strengthening of polyfunctionality:

Proposition 7.14

P is a polyfunction $\Leftarrow$ *inert*.P

Proof:

$$
\begin{array}{ll}
 & \exists(\, \alpha \;::\; \forall(\, t \;::\; \mathit{eqar} \sqcap P \circ {}^{<}t \circ P^{\cup} \sqsubseteq {}^{<}(\alpha \bullet t)\,)\,) \\
\Rightarrow & \quad \{ \text{ Proposition 6.11a; Proposition 7.5 } \} \\
 & \exists(\, \alpha \;::\; \forall(\, t \;::\; \mathit{eqar} \sqcap P \circ P^{\cup} \sqsubseteq {}^{<}t\,)\,) \\
= & \quad \{ \text{ predicate calculus } \} \\
 & \forall(\, t \;::\; \mathit{eqar} \sqcap P \circ P^{\cup} \sqsubseteq {}^{<}t\,) \\
= & \quad \{ \text{ plat calculus } \} \\
 & \mathit{eqar} \sqcap P \circ P^{\cup} \sqsubseteq \sqcap(\, t \;::\; {}^{<}t\,) \\
= & \quad \{ \text{ Proposition 6.14a } \} \\
 & \mathit{eqar} \sqcap P \circ P^{\cup} \sqsubseteq I \\
= & \quad \{ \text{ Definition 7.11 } \} \\
 & P \text{ is a polyfunction}
\end{array}
$$

□

There seems to be a problem: the translation of the intuitive idea of having a delay implies that the proc is a polyfunction. This excludes, for example, that an *arbitrary* delay is inert, whereas is it clear that such a delay only bases its present output on past input. The solution we propose is the following: first define causality for polyfunctions, and thereafter, extend this definition to arbitrary procs by requiring that causal procs can be decomposed into causal polyfunctions.
Next, we concentrate on a fundamental property of inert procs. In the literature, the feedback operator is sometimes called the fixpoint operator because it forces (part of) the output to be equal to (part of) the input. This motivates our interest in fixpoints of procs. In the model, a chronicle f is a fixpoint of proc P if and only if $f \langle P \rangle f$. In terms of relational algebra the fixpoints are represented by the interface $P \sqcap I$. The following abbreviation is used:

Definition 7.15 *Fixpoints*

$\mu P \triangleq P \sqcap I$

□

For an inert proc we can show that it has at most one fixpoint for every output arity: given an inert proc P, for any two chronicles f and g with the same arity such that $f \in \mu P$ and $g \in \mu P$ it follows that $f = g$. In relational algebra this translates to $\mu P \circ eqar \circ \mu P \sqsubseteq I$:

Proposition 7.16 *At most one fixpoint for every arity*

$$\mu P \circ eqar \circ \mu P \sqsubseteq I \;\Leftarrow\; inert.P$$

Proof:
From the characterisation of $inert.P$, 7.13, one obtains that there exists an archimedean function α such that:

$$\begin{array}{ll}
& \forall(\, t \;::\; eqar \sqcap P \circ {}^{<}t \circ P^{\cup} \;\sqsubseteq\; {}^{<}(\alpha \bullet t)\,) \\
\Rightarrow & \quad \{\; \mu P \sqsubseteq P \;\} \\
& \forall(\, t \;::\; eqar \sqcap \mu P \circ {}^{<}t \circ (\mu P)^{\cup} \;\sqsubseteq\; {}^{<}(\alpha \bullet t)\,) \\
= & \quad \{\; \mu P \sqsubseteq I\text{: Theorem 3.26a; } {}^{<}t \sqsubseteq eqar\text{: plat calculus} \;\} \\
& \forall(\, t \;::\; \mu P \circ {}^{<}t \circ \mu P \sqsubseteq {}^{<}(\alpha \bullet t)\,) \\
\Rightarrow & \quad \{\; \text{compose with } \mu P\text{; Corollary 3.27a} \;\} \\
& \forall(\, t \;::\; \mu P \circ {}^{<}t \circ \mu P \sqsubseteq \mu P \circ {}^{<}(\alpha \bullet t) \circ \mu P\,) \\
= & \quad \{\; \text{induction} \;\} \\
& \forall(\, t \;::\; \forall(\, n \;::\; \mu P \circ {}^{<}t \circ \mu P \;\sqsubseteq\; \mu P \circ {}^{<}(\alpha^n \bullet t) \circ \mu P\,)\,) \\
\Rightarrow & \quad \{\; \mu P \sqsubseteq I \;\} \\
& \forall(\, t \;::\; \forall(\, n \;::\; \mu P \circ {}^{<}t \circ \mu P \sqsubseteq {}^{<}(\alpha^n \bullet t)\,)\,) \\
= & \quad \{\; \text{plat calculus} \;\} \\
& \forall(\, t \;::\; \mu P \circ {}^{<}t \circ \mu P \;\sqsubseteq\; \sqcap(\, n \;::\; {}^{<}(\alpha^n \bullet t)\,)\,) \\
= & \quad \{\; \text{Proposition 7.6} \;\} \\
& \forall(\, t \;::\; \mu P \circ {}^{<}t \circ \mu P \sqsubseteq I\,) \\
= & \quad \{\; \text{plat calculus} \;\} \\
& \sqcup(\, t \;::\; \mu P \circ {}^{<}t \circ \mu P\,) \sqsubseteq I \\
= & \quad \{\; \text{cupjunctivity of composition} \;\} \\
& \mu P \circ \sqcup(\, t \;::\; {}^{<}t\,) \circ \mu P \;\sqsubseteq\; I \\
= & \quad \{\; \text{Proposition 6.14b} \;\} \\
& \mu P \circ eqar \circ \mu P \sqsubseteq I
\end{array}$$

□

A second problem is the non-preservation of totality, due to the fact that the domain can be incorrectly typed or incomplete. In Sections 7.1 we explained the problems. Now, we can suggest a solution. To capture the notion that for all sequences of chronicles in some domain the limit of that sequence is also in the domain, a new property *closedness* is defined:

Characterisation 7.17 *Closedness*

$$\begin{array}{ll}
& closed.P \\
\equiv & \\
& P \neq \perp\!\!\!\perp \;\wedge\; \sqcap(\, t \;::\; \top\!\!\top \circ P \circ {}^{<}t\,) \sqsubseteq \top\!\!\top \circ P
\end{array}$$

□

The proc $\perp\!\!\!\perp$ is excluded from the set of closed procs to avoid the occurrence of the conjunct $P \neq \perp\!\!\!\perp$ in almost all the important propositions to be proved in the sequel. Notice that the inclusion in the second conjunct is actually an equality: because of ${}^{<}t \sqsupseteq I$ the other inclusion follows.
Characterisation 7.17 is best understood in terms of interfaces A: a consequence of relational calculus is $closed.P \;\equiv\; closed.(P{>})$, and (right) domains are interfaces:

Proposition 7.18

$$closed.P \;\equiv\; closed.(P{>})$$

□

This follows from the Cone Rule 3.19 and Theorem 3.30c. In the model, Characterisation 7.17 for non-empty interface A reads:

$$\forall(\, t \;::\; \exists(\, g \;:\; g \in A \;:\; g \bullet (<\bullet t){\lhd} = f \bullet (<\bullet t){\lhd}\,)\,) \;\Rightarrow\; f \in A$$

We say that A is closed under limits. In particular, for an interface containing arbitrary long but finite chronicles of 1's, closedness guarantees that the limit of the finite chronicles (the chronicle carrying an infinite number of 1's) is also in the interface. The existence of limits in addition to being inert implies the existence of fixpoints. This is the counterpart of Proposition 7.16:

Proposition 7.19 *At least one fixpoint*

$$\mu P \neq \perp\!\!\!\perp \;\Leftarrow\; P \in A' \rightsquigarrow A' \;\wedge\; inert.P \;\wedge\; closed.P$$

Proof:
The proof is given in the model. First, from $P \in A' \rightsquigarrow A'$ we conclude, by Lemma 7.10a, $P \sqsubseteq eqar$, which implies, by Theorem 7.12, that P is a function. We will construct a sequence $(f_i)_i$ in A and show that the limit f_{lim} is *a* fixpoint of P, i.e., fulfills the equality $f_{lim} = P.f_{lim}$. The notation $P.f$ represents the unique chronicle returned by function P for $f \in A$.
From the primed typing information and $closed.P$ we get $P \sqcap eqar \neq \perp\!\!\!\perp$. This results in an initial value of a sequence:

$$\begin{array}{ll}
 & closed.P \\
\Rightarrow & \quad \{\ \text{Characterisation 7.17}\ \} \\
 & P \neq \perp\!\!\!\perp \\
= & \quad \{\ \text{Lemma 7.10a: } P \sqsubseteq eqar\ \} \\
 & P \sqcap eqar \neq \perp\!\!\!\perp \\
= & \quad \{\ \text{interpretation in the model}\ \} \\
 & \exists(\, f, g \;::\; f \,\langle P \rangle\, g \wedge f \,\langle eqar \rangle\, g\,) \\
= & \quad \{\ P \text{ is a function; Proposition 6.14b: interpretation in the model}\ \} \\
 & \exists(\, f, g \;::\; f = P.g \wedge \exists(\, t \;::\; f \,\langle {}^{<}t \rangle\, g\,)\,) \\
= & \quad \{\ \text{predicate calculus}\ \} \\
 & \exists(\, g, t \;::\; P.g \,\langle {}^{<}t \rangle\, g\,)\,)
\end{array}$$

There may be many chronicles g (with different arities) and moments t that fulfill the above condition. Let f_0 and t_0 be such that $P.f_0 \langle {}^{<}t_0 \rangle f_0$ holds. The chronicle f_0 is the initial value of the sequence $(f_i)_i$ which is constructed by iterating the application of P: $f_{n+1} \triangleq P.f_n$. This sequence exists in A because $P \in A \rightsquigarrow A$, and is unique, given the value f_0, because P is a function. Observe that f_0 fixes the arity of the sequence. The successive iterations are more and more alike. Let α witness the inertia of P; then:

$$\forall(\ n\ ::\ f_{n+1} \langle {}^{<}(\alpha^n \bullet t_0) \rangle\ f_n\ )$$

This is shown by the principle of induction. For the basis we calculate:

$$\begin{array}{ll} & f_1 \langle {}^{<}t_0 \rangle\ f_0 \\ = & \quad \{\ \text{definition } f_n\text{: } f_1 = P.f_0\ \} \\ & P.f_0 \langle {}^{<}t_0 \rangle\ f_0 \\ = & \quad \{\ \text{assumptions on } f_0 \text{ and } t_0\ \} \\ & \mathit{True} \end{array}$$

And for the step:

$$\begin{array}{ll} & f_{n+2} \langle {}^{<}(\alpha^{n+1} \bullet t_0) \rangle\ f_{n+1} \\ = & \quad \{\ \text{Definition 3.7b and } f_n\ \} \\ & P.f_{n+1} \langle {}^{<}(\alpha \bullet \alpha^n \bullet t_0) \rangle\ P.f_n \\ \Leftarrow & \quad \{\ f_{n+1}, f_n \in A;\ P \text{ is inert, witnessed by } \alpha\ \} \\ & f_{n+1} \langle {}^{<}(\alpha^n \bullet t_0) \rangle\ f_n \end{array}$$

Because α is unbounded, Proposition 7.4, we can define the limit of this sequence $(f_i)_i$. The limit is characterised by the following property:

$$\begin{array}{ll} & f_{lim} = g \\ \equiv & \\ & \forall(\ n\ ::\ g \langle {}^{<}(\alpha^n \bullet t_0) \rangle\ f_n\ ) \end{array}$$

The chronicle f_{lim} is a candidate for μP: $f_{lim} = P.f_{lim}$. This is proved by showing that $P.f_{lim}$ satisfies the characterising property of f_{lim}. First, it has to be assumed that f_{lim} is in A, the domain of P, but this is exactly described by $\sqcap(\ t\ ::\ \top\!\!\top \circ P \circ {}^{<}t\ ) \sqsubseteq \top\!\!\top \circ P$ in $\mathit{closed}.P$.

$$\begin{array}{ll} & f_{lim} = P.f_{lim} \\ = & \quad \{\ \text{Definition } f_{lim}\ \} \\ & \forall(\ n\ ::\ P.f_{lim} \langle {}^{<}(\alpha^n \bullet t_0) \rangle\ f_n\ ) \end{array}$$

Finally, by case analysis:
$n = 0$:

$$\begin{array}{ll} & P.f_{lim} \langle {}^{<}(\alpha^0 \bullet t_0) \rangle\ f_0 \\ = & \quad \{\ \text{Definition 3.7a; identity}\ \} \\ & P.f_{lim} \langle {}^{<}t_0 \rangle\ f_0 \\ \Leftarrow & \quad \{\ \text{Proposition 6.11b: } {}^{<}t \text{ is transitive}\ \} \\ & P.f_{lim} \langle {}^{<}t_0 \rangle\ P.f_0 \wedge P.f_0 \langle {}^{<}t_0 \rangle\ f_0 \end{array}$$

$$
\begin{array}{ll}
= & \{ \text{ assumptions on } f_0 \text{ and } t_0 \ \} \\
& P.f_{lim} \langle {<} t_0 \rangle \, P.f_0 \\
\Leftarrow & \{ \text{ Proposition 7.5 } \} \\
& P.f_{lim} \langle {<} (\alpha \bullet t_0) \rangle \, P.f_0 \\
\Leftarrow & \{ \ f_{lim}, f_0 \in A; \ P \text{ is inert, witnessed by } \alpha \ \} \\
& f_{lim} \langle {<} t_0 \rangle \, f_0 \\
= & \{ \ n = 0 \text{: characterisation } f_{lim} \ \} \\
& \mathit{True}
\end{array}
$$

$n > 0$:

$$
\begin{array}{ll}
& P.f_{lim} \langle {<} (\alpha^n \bullet t_0) \rangle \, f_n \\
= & \{ \ n > 0 \text{: Definition 3.7b and } f_n \ \} \\
& P.f_{lim} \langle {<} (\alpha \bullet \alpha^{n-1} \bullet t_0) \rangle \, P.f_{n-1} \\
\Leftarrow & \{ \ f_{lim}, f_{n-1} \in A; \ P \text{ is inert, witnessed by } \alpha \ \} \\
& f_{lim} \langle {<} (\alpha^{n-1} \bullet t_0) \rangle \, f_{n-1} \\
= & \{ \ n > 0 \text{: characterisation } f_{lim} \ \} \\
& \mathit{True}
\end{array}
$$

This concludes the proof of the existence of fixpoints.
□

Propositions 7.16 and 7.19 stress that a proc which is inert *and* closed has interesting properties; we had better combine the two notions into one: causality. This property specifies a class of processes that will turn out very important.

Characterisation 7.20 *Causality*

$$
\begin{array}{ll}
& \mathit{caus}.P \\
\equiv & \\
& \mathit{inert}.P \wedge \mathit{closed}.P
\end{array}
$$

□

Consequently, the main result of this investigation is:

Proposition 7.21 *Existence and uniqueness of fixpoint*

$$\mu P \neq \perp\!\!\!\perp \ \wedge \ \mu P \circ \mathit{eqar} \circ \mu P \sqsubseteq I \quad \Leftarrow \quad P \in A' \rightsquigarrow A' \ \wedge \ \mathit{caus}.P$$

□

Causality does not require a *well-defined arity*, despite the goal to describe implementability. This is done deliberately: when calculating with causal procs, we want to be as general as possible. This includes the irrelevance of the arities. After having finished the calculation, arities can be specialised to particular instances. Then we can conclude by Theorem 7.12 that we derived non-clairvoyant functions with a well-defined (closed) type.

Next, the main objective for introducing causality is shown: polyfunctionality and totality are preserved by feedback if the argument is causal. But first, the concept of *sectioning* is explained. Now, the use of points and extensionality enters the picture. In the lemmas

and propositions that follow, the formulation $P \circ x \triangle I$ appears. These forms are called *sections* of proc P; in this case it is a *left* section. The structure is known in functional programming as currying. For example, the binary function *Add* that adds two numbers has as a curried form (left section) the unary function *Add* x that increments its argument by x. The corresponding relational formulation reads $Add \circ x \triangle I$.
The lemma below shows what happens when P^σ is applied to some input x. Here, x is a point and can be thought of as the representation in the algebra $\mathcal{A}$ of a chronicle f in $\mathcal{B}$. The lemma states that the feedback construction gives as result the fixpoint(s) of the section $P \circ x \triangle I$:

Lemma 7.22

$$\forall(\, x \;::\; P^\sigma \circ x \;=\; \mu(P \circ x \triangle I) \circ \top\!\!\top \,)$$

Proof:
In the proof, the property is used that $x \triangle I$ is a function. This follows from the functionality of x (Definition 3.47) and I (Theorem 3.40a) and the functionality preservation of $\triangle$ (Theorem 3.36):

$$\begin{array}{ll} & P^\sigma \circ x \\ = & \quad \{\ \text{Definition 3.14}\ \} \\ & (P \sqcap \gg) \circ I \triangle \top\!\!\top \circ x \\ = & \quad \{\ \text{Corollary 3.24b}\ \} \\ & (P \sqcap \gg) \circ x \triangle \top\!\!\top \\ = & \quad \{\ \text{Theorem 3.48b}\ \} \\ & (P \sqcap \gg) \circ x \triangle I \circ \top\!\!\top \\ = & \quad \{\ x \triangle I \text{ is a function: Theorem 3.35a}\ \} \\ & (P \circ x \triangle I \;\sqcap\; \gg \circ x \triangle I) \circ \top\!\!\top \\ = & \quad \{\ \text{Split computation 3.13b}\ \} \\ & (P \circ x \triangle I \;\sqcap\; I \;\sqcap\; \top\!\!\top \circ x) \circ \top\!\!\top \\ = & \quad \{\ \text{Theorem 3.48c}\ \} \\ & (P \circ x \triangle I \;\sqcap\; I) \circ \top\!\!\top \\ = & \quad \{\ \text{Definition 7.15}\ \} \\ & \mu(P \circ x \triangle I) \circ \top\!\!\top \end{array}$$

□

The lemma above points out that the left section $P \circ x \triangle I$ is an interesting one. Sectioning preserves several of the properties introduced in the previous sections: primed typing, inertia, closedness and causality. Only the property of closedness requires a preliminary result. The lemma that follows shows that (left) sectioning preserves primed typing and totality of the argument proc:

Lemma 7.23

$$\forall(\, x \;:\; x \in B \;:\; P \circ x \triangle I \;\in\; A' \rightsquigarrow A' \,) \;\Leftarrow\; P \;\in\; A' \rightsquigarrow B \parallel A'$$

Proof:
By $x \in B$ we mean $B \circ x = x$. According to Theorem 3.39a and Lemma 7.10b, the primed typing of P implies: $P^{>} = B \parallel A$ and $P \sqsubseteq A \circ \mathit{eqar} \circ \gg \circ B \parallel A$. These properties are exploited in the proof:

$$
\begin{array}{ll}
 & P \circ x \vartriangle I \ \in \ A' \rightsquigarrow A' \\
= & \quad \{ \text{ Definition 3.38a } \} \\
 & P \circ x \vartriangle I \ \in \ A' \rightsquigarrow A' \ \wedge \ A \sqsubseteq (P \circ x \vartriangle I)^{>} \\
= & \quad \{ \text{ Lemma 7.10a; } (R \circ S)^{>} = (R^{>} \circ S)^{>} \ \} \\
 & P \circ x \vartriangle I \sqsubseteq A \circ \mathit{eqar} \circ A \ \wedge \ A \sqsubseteq (P^{>} \circ x \vartriangle I)^{>} \\
\Leftarrow & \quad \{ \text{ assumptions on } P \ \} \\
 & A \circ \mathit{eqar} \circ \gg \circ B \parallel A \circ x \vartriangle I \sqsubseteq A \circ \mathit{eqar} \circ A \ \wedge \ A \sqsubseteq (B \parallel A \circ x \vartriangle I)^{>} \\
= & \quad \{ \text{ Parallel-split fusion 3.12b } \} \\
 & A \circ \mathit{eqar} \circ \gg \circ (B \circ x) \vartriangle A \sqsubseteq A \circ \mathit{eqar} \circ A \ \wedge \ A \sqsubseteq ((B \circ x) \vartriangle A)^{>} \\
= & \quad \{ \ B \circ x = x \text{; Theorem 3.32b; } (P \vartriangle Q)^{>} = P^{>} \circ Q^{>} \ \} \\
 & A \circ \mathit{eqar} \circ A \circ x^{>} \sqsubseteq A \circ \mathit{eqar} \circ A \ \wedge \ A \sqsubseteq x^{>} \circ A \\
= & \quad \{ \text{ Definition 3.47 and Theorem 3.39b: } x^{>} = I \ \} \\
 & \mathit{True}
\end{array}
$$

□

This lemma is one step towards the application of Proposition 7.21 for $P \circ x \vartriangle I$. But more properties are required. The next proposition shows that (left) sectioning preserves inertia:

Proposition 7.24

$$\forall (\, x \ :: \ \mathit{inert}.(P \circ x \vartriangle I)\,) \ \Leftarrow \ \mathit{inert}.P$$

Proof:
It is shown that the archimedean function α witnessing the inertia of P also witnesses the inertia of $P \circ x \vartriangle I$:

$$
\begin{array}{ll}
 & \mathit{eqar} \sqcap P \circ x \vartriangle I \circ {}^{<}t \circ (P \circ x \vartriangle I)^{\cup} \\
= & \quad \{ \text{ Theorem 3.48b and Parallel-split fusion 3.12a; reverse } \} \\
 & \mathit{eqar} \sqcap P \circ I \parallel {}^{<}t \circ x \vartriangle I \circ (x \vartriangle I)^{\cup} \circ P^{\cup} \\
\sqsubseteq & \quad \{ \ R \vartriangle I \circ (S \vartriangle I)^{\cup} \sqsubseteq (R \circ S^{\cup}) \parallel I \ \} \\
 & \mathit{eqar} \ \sqcap \ P \ \circ \ I \parallel {}^{<}t \ \circ \ (x \circ x^{\cup}) \parallel I \ \circ \ P^{\cup} \\
= & \quad \{ \text{ Theorem 3.48a; Parallel-parallel fusion 3.12b } \} \\
 & \mathit{eqar} \sqcap P \circ x^{<} \parallel {}^{<}t \circ P^{\cup} \\
\sqsubseteq & \quad \{ \text{ Theorem 3.30d and Proposition 6.11a: } x^{<} \sqsubseteq I \sqsubseteq {}^{<}t \ \} \\
 & \mathit{eqar} \sqcap P \circ {}^{<}t \parallel {}^{<}t \circ P^{\cup} \\
\sqsubseteq & \quad \{ \text{ Proposition 6.11e } \} \\
 & \mathit{eqar} \sqcap P \circ {}^{<}t \circ P^{\cup} \\
\sqsubseteq & \quad \{ \ \alpha \text{ witnesses the inertia of } P \text{: Characterisation 7.13 } \} \\
 & {}^{<}(\alpha \bullet t)
\end{array}
$$

□

To prove the preservation of closedness, a difficult intermediate result is needed. It describes that P and Q are closed under limits whenever $P \parallel Q$ is closed. Fortunately, the applicability of this result is not restricted to this chapter only.

Proposition 7.25

$$\begin{array}{ll} & \mathit{closed}.P \wedge \mathit{closed}.Q \\ \Leftarrow & \\ & \mathit{closed}.(P \parallel Q) \end{array}$$

Proof:
The non-emptiness of P and Q follows straightforwardly from the non-emptiness of $P \parallel Q$ by Theorem 3.20c. For the second conjunct of the definition of closedness, Characterisation 7.17, the result following from relational calculus

$$\ll \circ R \parallel S \circ I^{\triangle} \top\!\!\top \;=\; R \;\Leftarrow\; S \neq \bot\!\!\bot$$

will be used. This result gives a way to introduce or eliminate $R \parallel S$. Now, $\mathit{closed}.P$ is derived:

$$\begin{array}{ll} & \top\!\!\top \circ P \\ = & \quad \{\; Q \neq \bot\!\!\bot \text{: above result} \;\} \\ & \top\!\!\top \circ \ll \circ P \parallel Q \circ I^{\triangle} \top\!\!\top \\ = & \quad \{\; \text{Theorem 3.40e: } \top\!\!\top \circ \ll \;=\; \top\!\!\top \circ I \parallel I \;\} \\ & \top\!\!\top \circ P \parallel Q \circ I^{\triangle} \top\!\!\top \\ \sqsupseteq & \quad \{\; \text{assumption on } P \parallel Q\text{: Characterisation 7.17} \;\} \\ & \sqcap(\, t \;::\; \top\!\!\top \circ P \parallel Q \circ {}^{<}t \,) \circ I^{\triangle} \top\!\!\top \\ = & \quad \{\; \text{Proposition 6.11d} \;\} \\ & \sqcap(\, t \;::\; \top\!\!\top \circ P \parallel Q \circ {}^{<}t \parallel {}^{<}t \,) \circ I^{\triangle} \top\!\!\top \\ = & \quad \{\; \text{Theorem 3.40e; Parallel-parallel fusion 3.12b} \;\} \\ & \sqcap(\, t \;::\; \top\!\!\top \circ \ll \circ (P \circ {}^{<}t) \parallel (Q \circ {}^{<}t) \,) \circ I^{\triangle} \top\!\!\top \\ = & \quad \{\; \text{Parallel computation 3.32c} \;\} \\ & \sqcap(\, t \;::\; \top\!\!\top \circ P \circ {}^{<}t \circ \ll \circ I \parallel (Q \circ {}^{<}t) \,) \circ I^{\triangle} \top\!\!\top \\ \sqsupseteq & \quad \{\; \text{Proposition 6.11a} \;\} \\ & \sqcap(\, t \;::\; \top\!\!\top \circ P \circ {}^{<}t \circ \ll \circ I \parallel Q \,) \circ I^{\triangle} \top\!\!\top \\ \sqsupseteq & \quad \{\; \text{monotonicity} \;\} \\ & \sqcap(\, t \;::\; \top\!\!\top \circ P \circ {}^{<}t \,) \circ \ll \circ I \parallel Q \circ I^{\triangle} \top\!\!\top \\ = & \quad \{\; Q \neq \bot\!\!\bot \text{: above result} \;\} \\ & \sqcap(\, t \;::\; \top\!\!\top \circ P \circ {}^{<}t \,) \end{array}$$

□

The main result of Proposition 7.25 and Lemma 7.23 is that closedness is preserved by sectioning. Despite the fact that more is assumed about P in Proposition 7.26, a sufficient condition is already that the right domain of P can be written as a *square* $B \parallel A$ for $\mathit{closed}.A$:

Proposition 7.26

$$\begin{array}{ll} & \forall(\, x \;:\; x \in B \;:\; closed.(P \circ x \triangle I)\,) \\ \Leftarrow & \\ & P \;\in\; A' \rightsquigarrow B \parallel A' \;\wedge\; closed.P \end{array}$$

Proof:

$$\begin{array}{ll} & closed.(P \circ x \triangle I) \\ = & \quad \{\ \text{Proposition 7.18}\ \} \\ & closed.((P \circ x \triangle I)_{>}) \\ \Leftarrow & \quad \{\ \text{Theorem 3.39a}\ \} \\ & P \circ x \triangle I \;\in\; A' \rightsquigarrow A' \;\wedge\; closed.A \\ \Leftarrow & \quad \{\ x \in B\text{: Lemma 7.23; Proposition 7.25}\ \} \\ & P \;\in\; A' \rightsquigarrow B \parallel A' \;\wedge\; closed.(B \parallel A) \\ = & \quad \{\ \text{Theorem 3.39a and Proposition 7.18}\ \} \\ & P \;\in\; A' \rightsquigarrow B \parallel A' \;\wedge\; closed.P \end{array}$$

□

These were all preliminary results, leading to the main preservation result. Under reasonable typing conditions, causality is preserved by sectioning:

Proposition 7.27

$$\begin{array}{ll} & \forall(\, x \;:\; x \in B \;:\; caus.(P \circ x \triangle I)\,) \\ \Leftarrow & \\ & P \;\in\; A' \rightsquigarrow B \parallel A' \;\wedge\; caus.P \end{array}$$

Proof:

$$\begin{array}{ll} & caus.(P \circ x \triangle I) \\ = & \quad \{\ \text{Characterisation 7.20}\ \} \\ & inert.(P \circ x \triangle I) \;\wedge\; closed.(P \circ x \triangle I) \\ \Leftarrow & \quad \{\ x \in B\text{: Propositions 7.24 and 7.26}\ \} \\ & P \;\in\; A' \rightsquigarrow B \parallel A' \;\wedge\; inert.P \;\wedge\; closed.P \\ = & \quad \{\ \text{Characterisation 7.20}\ \} \\ & P \;\in\; A' \rightsquigarrow B \parallel A' \;\wedge\; caus.P \end{array}$$

□

This property, together with Proposition 7.21, is the instrument to get a totality result for P^{σ}. First, we will concern ourselves with the polyfunctionality of the feedback.

7.4.1 Polyfunctionality of feedback

According to Characterisation 7.20 and Proposition 7.14, causality of P implies polyfunctionality. It is even the case that the feedback of a causal proc P is polyfunctional. In the presence of polymorphism, this is the best we can get: we saw that pure functionality

is not preserved by feedback, but polyfunctionality *is* preserved if the argument proc is causal:

Proposition 7.28

$$P^\sigma \text{ is a polyfunction} \Leftarrow \mathit{caus}.P$$

Proof:

$$\begin{array}{ll} & P^\sigma \text{ is a polyfunction} \\ = & \quad \{ \text{ Definition 7.11 } \} \\ & \mathit{eqar} \sqcap P^\sigma \circ (P^\sigma)^\cup \sqsubseteq I \\ = & \quad \{ \text{ Extensionality } \} \\ & \forall (\, x \;::\; \mathit{eqar} \sqcap P^\sigma \circ x \circ (P^\sigma \circ x)^\cup \sqsubseteq I \,) \end{array}$$

We continue: for all x,

$$\begin{array}{ll} & \mathit{eqar} \sqcap P^\sigma \circ x \circ (P^\sigma \circ x)^\cup \sqsubseteq I \\ = & \quad \{ \text{ Lemma 7.22 } \} \\ & \mathit{eqar} \sqcap \mu(P \circ x \triangle I) \circ \top\!\!\top \circ (\mu(P \circ x \triangle I) \circ \top\!\!\top)^\cup \sqsubseteq I \\ = & \quad \{ \text{ reverse through composition } \} \\ & \mathit{eqar} \sqcap \mu(P \circ x \triangle I) \circ \top\!\!\top \circ \top\!\!\top^\cup \circ (\mu(P \circ x \triangle I))^\cup \sqsubseteq I \\ = & \quad \{ \top\!\!\top \circ \top\!\!\top^\cup = \top\!\!\top;\ \mu P \text{ is an interface: Theorem 3.26a } \} \\ & \mathit{eqar} \sqcap \mu(P \circ x \triangle I) \circ \top\!\!\top \circ \mu(P \circ x \triangle I) \sqsubseteq I \\ = & \quad \{ \mu(P \circ x \triangle I) \text{ is an interface; } Q \sqcap A \circ \top\!\!\top \circ A = A \circ Q \circ A \ \} \\ & \mu(P \circ x \triangle I) \circ \mathit{eqar} \circ \mu(P \circ x \triangle I) \sqsubseteq I \\ \Leftarrow & \quad \{ \text{ Proposition 7.16 } \} \\ & \mathit{inert}.(P \circ x \triangle I) \\ \Leftarrow & \quad \{ \text{ Proposition 7.24 } \} \\ & \mathit{inert}.P \\ \Leftarrow & \quad \{ \text{ Characterisation 7.20 } \} \\ & \mathit{caus}.P \end{array}$$

□

Observe that only the inert part of P was used to get the result. This is not very surprising, since the property of closedness was introduced to preserve totality. This is the topic of the subsequent subsection.

7.4.2 Totality of feedback

In this subsection a rule is given which allows to conclude the totality of a feedback structure on some interface. The assumption that the argument proc P for feedback is closed under limits, Characterisation 7.17, is essential. First, a weaker version of preservation of typing is proved:

Lemma 7.29

$$P^\sigma \in A \sim B \;\Leftarrow\; P \in A \sim B \parallel A$$

Proof:
According to Definition 3.28 and Theorem 3.31b the assumption on P translates to the expression $P = A \circ P \circ B \parallel A$. The proof obligation then reads $P^\sigma = A \circ P^\sigma \circ B$:

$$
\begin{array}{ll}
 & A \circ P^\sigma \circ B \\
= & \quad \{ \text{ Definition 3.14 } \} \\
 & A \circ (P \sqcap \gg) \circ I \vartriangle \top\!\!\top \circ B \\
= & \quad \{ \text{ Corollary 3.27c; Corollary 3.24b } \} \\
 & (A \circ P \sqcap A \circ \gg) \circ B \vartriangle \top\!\!\top \\
= & \quad \{ \text{ Parallel computation 3.32d; Parallel-split fusion 3.12a } \} \\
 & (A \circ P \sqcap \gg \circ I \parallel A) \circ B \parallel I \circ I \vartriangle \top\!\!\top \\
= & \quad \{ \text{ Corollary 3.27c; Parallel-parallel fusion 3.12b } \} \\
 & (A \circ P \sqcap \gg \circ B \parallel A) \circ I \vartriangle \top\!\!\top \\
= & \quad \{ \text{ Corollary 3.27c } \} \\
 & (A \circ P \circ B \parallel A \sqcap \gg) \circ I \vartriangle \top\!\!\top \\
= & \quad \{ \text{ assumption on } P \ \} \\
 & (P \sqcap \gg) \circ I \vartriangle \top\!\!\top \\
= & \quad \{ \text{ Definition 3.14 } \} \\
 & P^\sigma
\end{array}
$$

□

This paves the way to proving the totality of feedback if the argument is a total and causal polyfunction. Because the hard work has been done in Proposition 7.21 (and Proposition 7.19), the proof is extremely short:

Proposition 7.30

$$P^\sigma \in A \rightsquigarrow B \quad \Leftarrow \quad P \in A' \rightsquigarrow B \parallel A' \ \wedge \ \mathit{caus}.P$$

Proof:

$$
\begin{array}{ll}
 & P^\sigma \in A \rightsquigarrow B \\
= & \quad \{ \text{ Definitions 3.38a and 3.37a } \} \\
 & P^\sigma \in A \sim B \ \wedge \ B \sqsubseteq (P^\sigma)\rangle \\
= & \quad \{ \ P \in A \sim B \parallel A \text{: Lemma 7.29 } \} \\
 & B \sqsubseteq (P^\sigma)\rangle
\end{array}
$$

This last expression is, by an extensional argument, Axiom 3.49, and making use of the Cone Rule 3.19, equivalent to:

$$\forall(\, x \ : \ x \in B \ : \ P^\sigma \circ x \neq \perp\!\!\!\perp \,)$$

Now, the proof continues: for all $x \in B$,

$$
\begin{array}{ll}
 & P^\sigma \circ x \neq \perp\!\!\!\perp \\
= & \quad \{ \text{ Lemma 7.22 } \} \\
 & \mu(P \circ x \vartriangle I) \circ \top\!\!\top \ \neq \ \perp\!\!\!\perp \\
= & \quad \{ \text{ Cone Rule 3.19 and Theorem 3.6 } \}
\end{array}
$$

$$
\begin{array}{ll}
 & \mu(P \circ x \triangleleft I) \neq \bot\!\!\bot \\
\Leftarrow & \quad \{ \text{ Proposition 7.21 } \} \\
 & P \circ x \triangleleft I \in A' \rightsquigarrow A' \ \wedge \ caus.(P \circ x \triangleleft I) \\
\Leftarrow & \quad \{ \ x \in B\text{: Lemma 7.23 and Proposition 7.27 } \} \\
 & P \in A' \rightsquigarrow B \parallel A' \ \wedge \ caus.P
\end{array}
$$

□

These are all useful results but not general yet enough: because $caus.P$ implies polyfunctionality of P, we can not state results like Proposition 7.30 or 7.28 for non-polyfunctional procs. Therefore, we have to extend the definition of causality to arbitrary procs.

7.5 Causality for arbitrary procs

To define causality for arbitrary procs, advantage is taken of the view that a proc is the union of all the (poly)functional procs which are included in that proc and defined on the same domain[1]. The combined notion of 'included' and 'same domain' is sometimes dubbed *refinement*:

Definition 7.31 *Refinement*

$$
\begin{array}{ll}
 & P \leq Q \\
\equiv & \\
 & P \sqsubseteq Q \ \wedge \ P\rangle = Q\rangle
\end{array}
$$

□

$P \leq Q$ is pronounced as 'P refines Q'. In Back [Bac93], Hoare [Hoa85] and Morgan [Mor90] the direction of the refinement symbol is reversed. Moreover, the notion defined in those papers is somewhat weaker. The choice for the direction in the definition above originates from the order on procs, which is due to set inclusion in the model.

Theorem 7.32 *Refinement*

a. $\leq$ is a partial order

b. $P \leq F \ \equiv \ P = F$

□

Several operators from the relational algebra, such as $\sqcup$, $\parallel$ and $\triangleleft$, preserve refinement. Other operators, such as $\sqcap$, $\circ$, $\cup$ and $^\sigma$, do not preserve refinement, due to the domain requirement in Definition 7.31. Additional typing information (or even closedness information) of the arguments avoids this problem.

[1] This is an extensional argument.

Next, we extend Characterisation 7.20 to arbitrary procs:

Characterisation 7.33 *Causality*

$$\begin{array}{ll} & \mathit{causal}.P \\ \equiv & \\ & \perp\!\!\!\perp \neq P \sqsubseteq \sqcup(\, Q \;:\; Q \leq P \wedge \mathit{caus}.Q \;:\; Q \,) \end{array}$$

In combination with typing, the notation '$\overset{c}{\sim}$' is used to denote that the proc is causal
□

The non-emptiness of closed P is equivalent to the existence of a proc Q such that $Q \leq P$ and $\mathit{caus}.Q$. Observe that the inclusion in the definition is actually an equality. A healthiness condition is that Characterisation 7.33 is a real extension of Characterisation 7.20. It is even the case that the two are equivalent when applied to a function:

Proposition 7.34

a. $\mathit{causal}.F \equiv \mathit{caus}.F$

b. $\mathit{causal}.P \Leftarrow \mathit{caus}.P$

Proof:
We only show 7.34a:

$$\begin{array}{ll} & \mathit{causal}.F \\ = & \quad \{\ \text{Characterisation 7.33}\ \} \\ & \perp\!\!\!\perp \neq F \sqsubseteq \sqcup(\, Q \;:\; Q \leq F \wedge \mathit{caus}.Q \;:\; Q \,) \\ = & \quad \{\ \text{Theorem 7.32b; predicate calculus}\ \} \\ & \perp\!\!\!\perp \neq F \sqsubseteq \sqcup(\, Q \;:\; Q = F \wedge \mathit{caus}.Q \;:\; Q \,) \;\wedge\; (\mathit{caus}.F \vee \neg(\mathit{caus}.F)) \\ = & \quad \{\ \text{Leibniz; } \wedge \text{ distributes over } \vee\ \} \\ & (\perp\!\!\!\perp \neq F \sqsubseteq \sqcup(\, Q \;:\; Q = F \wedge \mathit{caus}.F \;:\; F \,) \;\wedge\; \mathit{caus}.F) \\ & \;\vee\; (\perp\!\!\!\perp \neq F \sqsubseteq \sqcup(\, Q \;:\; Q = F \wedge \mathit{caus}.F \;:\; F \,) \;\wedge\; \neg(\mathit{caus}.F)) \\ = & \quad \{\ \text{plat calculus}\ \} \\ & (\perp\!\!\!\perp \neq F \sqsubseteq F \wedge \mathit{caus}.F) \;\vee\; (\perp\!\!\!\perp \neq F \sqsubseteq \perp\!\!\!\perp \wedge \neg(\mathit{caus}.F)) \\ = & \quad \{\ \mathit{caus}.F \Rightarrow F \neq \perp\!\!\!\perp\text{; predicate calculus}\ \} \\ & \mathit{caus}.F \end{array}$$

□

The equivalence for functions of Proposition 7.34a does not hold for polyfunctions. This is due to the fact that the least archimedean function ('least' in the sense of the pointwise ordering) of an arbitrary set of archimedean functions does in general not exist.
The most important corollary of Propositions 7.34a and 7.28 is that the feedback of a causal function results in a polyfunction. Recall that a function is also a polyfunction, Theorem 7.12. We intend to axiomatise the result; that's why it is called a *property*:

Property 7.35 *Preservation of polyfunctionality*

$$F^{\sigma} \text{ is a polyfunction} \Leftarrow \mathit{causal}.F$$

□

With the extended definition of causality, Characterisation 7.33, we can take a closer look at a final property for totality preservation by feedback.

7.5.1 Totality of feedback revisited

The following property generalised Proposition 7.30. This is the second and last *property* of this chapter on causality:

Property 7.36 *Preservation of totality*

$$P^\sigma \in A \rightsquigarrow B \;\Leftarrow\; P \in A' \overset{c}{\rightsquigarrow} B \parallel A'$$

Proof:
In the proof, the fact is used that $Q \leq P$ and $P \in A' \rightsquigarrow B \parallel A'$ imply $Q \in A' \rightsquigarrow B \parallel A'$. This enables us to exploit Proposition 7.30:

$$\begin{array}{ll}
 & P^\sigma \in A \rightsquigarrow B \\
= & \quad \{ \text{ Definitions 3.38a and 3.37a } \} \\
 & P^\sigma \in A \sim B \;\wedge\; B \sqsubseteq (P^\sigma)_\triangleright \\
= & \quad \{ \; P \in A \sim B \parallel A \text{: Lemma 7.29 } \} \\
 & B \sqsubseteq (P^\sigma)_\triangleright \\
\Leftarrow & \quad \{ \; \sqcup(\, Q \;:\; Q \leq P \wedge \mathit{caus}.Q \;:\; Q \,) \sqsubseteq P \text{; monotonicity } \} \\
 & B \sqsubseteq (\sqcup(\, Q \;:\; Q \leq P \wedge \mathit{caus}.Q \;:\; Q \,)^\sigma)_\triangleright \\
= & \quad \{ \text{ cupjunctivity of feedback and domains } \} \\
 & B \sqsubseteq \sqcup(\, Q \;:\; Q \leq P \wedge \mathit{caus}.Q \;:\; (Q^\sigma)_\triangleright \,) \\
= & \quad \{ \; Q \in A' \rightsquigarrow B \parallel A' \text{: Proposition 7.30 and Theorem 3.39a } \} \\
 & B \sqsubseteq \sqcup(\, Q \;:\; Q \leq P \wedge \mathit{caus}.Q \;:\; B \,) \\
= & \quad \{ \text{ Characterisation 7.33: } \exists(\, Q \;::\; Q \leq P \wedge \mathit{caus}.Q \,), \text{ predicate calculus } \} \\
 & \mathit{True}
\end{array}$$

□

This concludes the discussion on the consequences of causality. The two properties in this chapter motivate the importance of causality, and consequently urges us to look at causality properties of the constants in the relational algebra and to investigate preservation of causality by several composition constructions. This is the main topic in the next chapter.

Chapter 8

Preservation of causality

This chapter covers two things. First, the causality of several constants from the relational algebra is proved. Secondly, the important composition constructions are considered with respect to their causality-preservation properties.

8.1 Constants

The non-causality of $\perp\!\!\!\perp$ follows immediately from the observation that this proc is not closed, Characterisation 7.17. The non-causality of I (and of $\ll$ and $\gg$) is a bit tricky. One can show that $causal.I$ is equivalent to $I = \top\!\top$. So, to conclude $\neg(causal.I)$, we need the axiom $I \neq \top\!\top$. However, one should keep in mind that at the end we want to drop the definition of causality, and continue with the derived properties such as causality preservation. Therefore, the non-causality of I is not important, and the required axiom is not imposed. Really important is to know when a proc *is* causal:

Property 8.1 *Causality of constants*

a. $causal.\top\!\top$

b. $causal.eqar$

Proof:
To prove causality of $\top\!\top$ we have to give a set of causal polyfunctions (in the sense of Characterisation 7.20) such that the cup over this set is $\top\!\top$. According to Theorem 3.50b, we can decompose $\top\!\top$ into points. Therefore, it suffices to show that all points are causal: $\forall(\, x \;::\; caus.x \,)$:

$$
\begin{array}{ll}
 & inert.x \\
= & \quad \{ \text{ Characterisation 7.13 } \} \\
 & \exists(\, \alpha \;::\; \forall(\, t \;::\; eqar \sqcap x \circ {}^<t \circ x^\cup \sqsubseteq {}^<(\alpha \bullet t)\,)\,) \\
\Leftarrow & \quad \{ \; {}^<t \sqsubseteq \top\!\top \text{ and Proposition 6.11a: monotonicity } \} \\
 & \exists(\, \alpha \;::\; \forall(\, t \;::\; x \circ \top\!\top \circ x^\cup \sqsubseteq I\,)\,) \\
= & \quad \{ \text{ predicate calculus } \}
\end{array}
$$

$$
\begin{array}{ll}
 & x \circ \top\!\!\top \circ x^{\cup} \sqsubseteq I \\
= & \quad \{ \text{ Definition 3.47: } x \circ \top\!\!\top = x \ \} \\
 & x \circ x^{\cup} \sqsubseteq I \\
= & \quad \{ \text{ Definition 3.47: a point is a function; Definition 3.33a } \} \\
 & \mathit{True}
\end{array}
$$

And for the property of closedness it is shown that points are closed under limits:

$$
\begin{array}{ll}
 & \mathit{closed}.x \\
= & \quad \{ \text{ Characterisation 7.17 } \} \\
 & x \neq \bot\!\!\bot \ \wedge \ \sqcap(\, t \ :: \ \top\!\!\top \circ x \circ {}^{<}t \,) \sqsubseteq \top\!\!\top \circ x \\
= & \quad \{ \text{ Cone Rule 3.19 } \} \\
 & \top\!\!\top \circ x \circ \top\!\!\top = \top\!\!\top \ \wedge \ \sqcap(\, t \ :: \ \top\!\!\top \circ x \circ {}^{<}t \,) \sqsubseteq \top\!\!\top \circ x \\
= & \quad \{ \text{ Theorem 3.48c } \} \\
 & \top\!\!\top \circ \top\!\!\top = \top\!\!\top \ \wedge \ \sqcap(\, t \ :: \ \top\!\!\top \circ {}^{<}t \,) \sqsubseteq \top\!\!\top \\
= & \quad \{ \text{ Theorem 3.6; } \top\!\!\top \text{ is top element } \} \\
 & \mathit{True}
\end{array}
$$

Straightforwardly, the causality of $\top\!\!\top$ follows from the causality of points:

$$
\begin{array}{ll}
 & \mathit{causal}.\top\!\!\top \\
= & \quad \{ \text{ Characterisation 7.33 } \} \\
 & \bot\!\!\bot \neq \top\!\!\top \sqsubseteq \sqcup(\, Q \ : \ Q \leq \top\!\!\top \wedge \ \mathit{caus}.Q \ : \ Q \,) \\
= & \quad \{ \text{ Theorem 3.20b; } Q \leq \top\!\!\top \ \equiv \ Q \in I \rightsquigarrow I \ \} \\
 & \top\!\!\top \sqsubseteq \sqcup(\, Q \ : \ Q \in I \rightsquigarrow I \ \wedge \ \mathit{caus}.Q \ : \ Q \,) \\
\Leftarrow & \quad \{ \text{ plat calculus } \} \\
 & \top\!\!\top = \sqcup(\, x \ : \ x \in I \leftarrowtail I \ \wedge \ \mathit{caus}.x \ : \ x \,) \\
= & \quad \{ \text{ Definition 3.47; points are causal } \} \\
 & \top\!\!\top = \sqcup(\, x \ :: \ x \,) \\
= & \quad \{ \text{ Theorem 3.50b } \} \\
 & \mathit{True}
\end{array}
$$

The proof of 8.1b has the same structure. From Theorem 5.13a, the non-emptiness of *eqar* is derived. To complete the proof, we need the notion of *polypoint*, which is a generalisation of a point in the following sense:

$$
\begin{array}{ll}
 & px \text{ is a polypoint} \\
\equiv & \\
 & \forall(\, X \ : \ X \in \mathcal{D} \ : \ X \circ px \text{ is a point} \,)
\end{array}
$$

Observe that polypoints are polyfunctions, just like points are functions. Because we can construct a pair (X, px) of every normal point, we conclude by Theorem 3.50b that $\sqcup(\, px \ :: \ px \,) = \top\!\!\top$. This implies that we can decompose *eqar* into $\sqcup(\, X \ :: \ X \circ px \circ X \,)$-structures, for all px. These structures turn out to be real refinements of *eqar*. Moreover, they are inert; this follows from the equivalence for all $X, Y \in \mathcal{D}$:

$$
\begin{array}{ll}
 & X \circ \mathit{eqar} \circ Y \neq \bot\!\!\bot \\
\equiv & \\
 & X = Y
\end{array}
$$

Because the structures are also total on I, which is closed, we conclude that for all X and px, $\sqcup(\, X \;::\; X \circ px \circ X \,)$ is causal in the sense of Characterisation 7.20:

$$
\begin{array}{ll}
 & \mathit{eqar} \\
= & \quad \{ \text{ Lemma 7.8b; } \sqcup(\, px \;::\; px \,) = \top\!\top \ \} \\
 & \sqcup(\, X \;::\; X \circ \sqcup(\, px \;::\; px \,) \circ X \,) \\
= & \quad \{ \text{ cupjunctivity } \} \\
 & \sqcup(\, px \;::\; \sqcup(\, X \;::\; X \circ px \circ X \,) \,) \\
= & \quad \{ \ \sqcup(\, X \;::\; X \circ px \circ X \,) \leq \mathit{eqar} \text{ and is causal } \} \\
 & \sqcup(\, px \;:\; \sqcup(\, X \;::\; X \circ px \circ X \,) \leq \mathit{eqar} \ \wedge \ \mathit{caus}.\sqcup(\, X \;::\; X \circ px \circ X \,) \\
 & \qquad :\; \sqcup(\, X \;::\; X \circ px \circ X \,) \,) \\
\sqsubseteq & \quad \{ \text{ monotonicity } \} \\
 & \sqcup(\, Q \;:\; Q \leq \mathit{eqar} \wedge \mathit{caus}.Q \;:\; Q \,)
\end{array}
$$

□

Despite the fact that the procs I, $\ll$ and $\gg$ are not causal, they still satisfy properties like closedness and not relating present output to future input. In Chapter 9 the notion of *weak causality* is introduced. The procs I, $\ll$ and $\gg$ turn out to be causal in this weaker sense.

8.2 Preservation

In this section an important aspect is handled: preservation of causality by several constructions such as cup, sequential composition, parallel composition and feedback.

8.2.1 Cup

The causality preservation of $\sqcup$ in the sense of Characterisation 7.20 requires a preliminary result: a minimum operator combines two archimedean functions into one. The proof that we indeed obtain an archimedean function is beyond the scope of this presentation; therefore, the exercise is placed in Appendix A.

Proposition 8.2

$$\mathit{caus}.(P \sqcup Q) \ \Leftarrow \ \mathit{caus}.P \ \wedge \ \mathit{caus}.Q \ \wedge \ P \circ \mathit{eqar} \circ Q^{\cup} = \bot\!\!\!\bot$$

Proof:
In Appendix A, the function $\downarrow$ is investigated. It is the minimum on moments. Its definition is:

$$\downarrow \ \triangleq \ (\mathcal{I} \blacktriangleleft \geqslant \ \cup \ \geqslant \blacktriangleleft \mathcal{I})^{-1}$$

In the model, the interpretation reads:

$$\forall(\, t, t', t'' \;::\; t \,\langle\downarrow\rangle\, (t', t'') \ \equiv \ (t = t' \ \wedge \ t \leqslant t'') \ \vee \ (t \leqslant t' \ \wedge \ t = t'') \,)$$

With this function, the preservation of inertia by cup can be proved; after that the proof for closedness is completed.

The assumption on P and Q is that they are inert, witnessed by some α and β, respectively. For all moments t:

$$
\begin{array}{ll}
 & \mathit{eqar} \sqcap (P \sqcup Q) \circ {}^{<}t \circ (P \sqcup Q)^{\cup} \\
= & \quad \{ \text{ cupjunctivity } \} \\
 & \mathit{eqar} \sqcap (P \circ {}^{<}t \circ P^{\cup} \sqcup P \circ {}^{<}t \circ Q^{\cup} \sqcup Q \circ {}^{<}t \circ P^{\cup} \sqcup Q \circ {}^{<}t \circ Q^{\cup}) \\
= & \quad \{ \ P \circ {}^{<}t \circ Q^{\cup} \sqsubseteq P \circ \mathit{eqar} \circ Q^{\cup} = \bot\!\!\!\bot \ \} \\
 & \mathit{eqar} \sqcap (P \circ {}^{<}t \circ P^{\cup} \sqcup Q \circ {}^{<}t \circ Q^{\cup}) \\
= & \quad \{ \text{ cupjunctivity } \} \\
 & (\mathit{eqar} \sqcap P \circ {}^{<}t \circ P^{\cup}) \sqcup (\mathit{eqar} \sqcap Q \circ {}^{<}t \circ Q^{\cup}) \\
\sqsubseteq & \quad \{ \text{ inertia of } P \text{ and } Q \ \} \\
 & {}^{<}(\alpha \bullet t) \sqcup {}^{<}(\beta \bullet t) \\
= & \quad \{ \text{ Proposition A.9 } \} \\
 & {}^{<}(\downarrow \bullet \alpha \blacktriangle \beta \bullet t)
\end{array}
$$

So inertia of $P \sqcup Q$ is witnessed by the archimedean function $\downarrow \bullet \alpha \blacktriangle \beta$, Proposition A.8. To prove that $P \sqcup Q$ is closed we do not need the assumption $P \circ \mathit{eqar} \circ Q^{\cup} = \bot\!\!\!\bot$. Non-emptiness of $P \sqcup Q$ is trivial because P and Q are both non-empty. A powerful step in the following proof is the diagonalisation rule:

$$
\begin{array}{ll}
 & \top\!\!\top \circ (P \sqcup Q) \\
= & \quad \{ \text{ cupjunctivity } \} \\
 & \top\!\!\top \circ P \sqcup \top\!\!\top \circ Q \\
\sqsupseteq & \quad \{ \ P \text{ and } Q \text{ are closed } \} \\
 & \sqcap(\ t \ :: \ \top\!\!\top \circ P \circ {}^{<}t \ ) \sqcup \sqcap(\ t' \ :: \ \top\!\!\top \circ Q \circ {}^{<}t' \ ) \\
= & \quad \{ \text{ capjunctivity of cup } \} \\
 & \sqcap(\ t, t' \ :: \ \top\!\!\top \circ P \circ {}^{<}t \sqcup \top\!\!\top \circ Q \circ {}^{<}t' \ ) \\
= & \quad \{ \text{ Proposition 6.13c: diagonalisation } \} \\
 & \sqcap(\ t \ :: \ \top\!\!\top \circ P \circ {}^{<}t \sqcup \top\!\!\top \circ Q \circ {}^{<}t \ ) \\
= & \quad \{ \text{ cupjunctivity } \} \\
 & \sqcap(\ t \ :: \ \top\!\!\top \circ (P \sqcup Q) \circ {}^{<}t \ )
\end{array}
$$

□

Causality in the sense of Characterisation 7.33 is proved more straightforwardly. Two rules are given: one for the union of procs with the same right domain, and one for the case when the two components have completely disjoint arities:

Proposition 8.3

a. $\mathit{causal}.(P \sqcup Q) \ \Leftarrow \ \mathit{causal}.P \wedge \mathit{causal}.Q \wedge P^{>} = Q^{>}$

b. $\mathit{causal}.(P \sqcup Q) \ \Leftarrow \ \mathit{causal}.P \wedge \mathit{causal}.Q \wedge P \circ \mathit{eqar} \circ Q^{\cup} = \bot\!\!\!\bot$

Proof of 8.3a:

$$
\begin{array}{ll}
 & \mathit{causal}.(P \sqcup Q) \\
= & \quad \{ \text{ Characterisation 7.33 } \}
\end{array}
$$

$$
\begin{array}{ll}
 & \bot\!\!\bot \neq P \sqcup Q \sqsubseteq \sqcup(\, R \;:\; R \leq P \sqcup Q \;\wedge\; caus.R \;:\; R \,) \\
= & \qquad \{\ \text{property cup}\ \} \\
 & \bot\!\!\bot \neq P \sqcup Q \;\wedge\; P \sqsubseteq \sqcup(\, R \;:\; R \leq P \sqcup Q \;\wedge\; caus.R \;:\; R \,) \\
 & \;\wedge\; Q \sqsubseteq \sqcup(\, R \;:\; R \leq P \sqcup Q \;\wedge\; caus.R \;:\; R \,) \\
\Leftarrow & \qquad \{\ P{>} = Q{>} \wedge R \leq P \;\Rightarrow\; R \leq P \sqcup Q\ \} \\
 & \bot\!\!\bot \neq P \;\wedge\; \bot\!\!\bot \neq Q \;\wedge\; P \sqsubseteq \sqcup(\, R \;:\; R \leq P \wedge caus.R \;:\; R \,) \\
 & \;\wedge\; Q \sqsubseteq \sqcup(\, R \;:\; R \leq Q \wedge caus.R \;:\; R \,) \\
= & \qquad \{\ \text{Characterisation 7.33}\ \} \\
 & causal.P \wedge causal.Q
\end{array}
$$

Proposition 8.3b follows from Proposition 8.2. In the proof the property

$$
\begin{array}{ll}
 & P' \sqcup Q' \leq P \sqcup Q \;\wedge\; P' \circ eqar \circ Q'^{\cup} = \bot\!\!\bot \\
\Leftarrow & \\
 & P' \leq P \;\wedge\; Q' \leq Q \;\wedge\; P \circ eqar \circ Q^{\cup} = \bot\!\!\bot
\end{array}
$$

is used

□

This proposition can be generalised to arbitrary cups of procs with the same domain, or with pairwise 'disjoint' domains. For non-empty set $\mathcal{S}$ of procs:

Property 8.4 *Cup*

a. $causal.(\sqcup\mathcal{S}) \;\Leftarrow\; \forall(\, P \;:\; P \in \mathcal{S} \;:\; P \in I \overset{c}{\rightsquigarrow} A \,)$

b. $causal.(\sqcup\mathcal{S})$

$\Leftarrow$

$\forall(\, P,Q \;:\; P,Q \in \mathcal{S} \;:\; causal.P \;\wedge\; (P \neq Q \;\Rightarrow\; P \circ eqar \circ Q^{\cup} = \bot\!\!\bot)\,)$

□

Property 8.4a suggests a mechanism to prove the causality of a non-deterministic proc. It is a direct corollary of Characterisation 7.33. In particular 8.4b is interesting because it captures the construction of polymorphic, causal procs.

8.2.2 Sequential composition

The preservation of causality by sequential composition is simple. Parallel composition will cause some more trouble.

Proposition 8.5

$$
\begin{array}{ll}
 & caus.(P \circ Q) \\
\Leftarrow & \\
 & caus.P \;\wedge\; caus.Q \;\wedge\; P{>} \sqsupseteq Q{<} \\
 & \;\wedge\; (P^{\cup} \circ eqar \circ P \sqsubseteq eqar \;\vee\; Q \circ eqar \circ Q^{\cup} \sqsubseteq eqar)
\end{array}
$$

First the property of inertia is proved; after that the proof for closedness is given. The assumption on P and Q is that they are inert, witnessed by some α and β, respectively.

The calculation below shows that α witnesses the inertia of $P \circ Q$. For all t:

$$
\begin{array}{ll}
 & eqar \sqcap P \circ Q \circ {}^{<}t \circ (P \circ Q)^{\cup} \\
= & \quad \{ \text{ reverse through composition } \} \\
 & eqar \sqcap P \circ Q \circ {}^{<}t \circ Q^{\cup} \circ P^{\cup} \\
= & \quad \{ \ P^{\cup} \circ eqar \circ P \sqsubseteq eqar \ \vee \ Q \circ eqar \circ Q^{\cup} \sqsubseteq eqar \ \} \\
 & eqar \ \sqcap \ P \ \circ \ (eqar \sqcap Q \circ {}^{<}t \circ Q^{\cup}) \ \circ \ P^{\cup} \\
\sqsubseteq & \quad \{ \text{ inertia of } Q \ \} \\
 & eqar \sqcap P \circ {}^{<}(\beta \bullet t) \circ P^{\cup} \\
\sqsubseteq & \quad \{ \text{ Proposition 7.5 } \} \\
 & eqar \sqcap P \circ {}^{<}t \circ P^{\cup} \\
\sqsubseteq & \quad \{ \text{ inertia of } P \ \} \\
 & {}^{<}(\alpha \bullet t)
\end{array}
$$

To prove that $P \circ Q$ is closed, observe that $(P \circ Q)^{>} = Q^{>}$ by the typing assumptions on P and Q:

$$
\begin{array}{ll}
 & (P \circ Q)^{>} \\
= & \quad \{ \ (P \circ Q)^{>} = (P^{>} \circ Q)^{>}; \text{ Theorem 3.30b } \} \\
 & (P^{>} \circ Q^{<} \circ Q)^{>} \\
= & \quad \{ \ I \sqsupseteq P^{>} \sqsupseteq Q^{<}: \text{ Corollary 3.27b and plat calculus } \} \\
 & (Q^{<} \circ Q)^{>} \\
= & \quad \{ \text{ Theorem 3.30b } \} \\
 & Q^{>}
\end{array}
$$

The fact that $P \circ Q$ is closed follows now from the corresponding property of Q and Proposition 7.18

□

The typing requirements in the antecedent of Proposition 8.5 are needed to ensure that proc Q does not produce output for which P is not defined. Observe that in this case, the angelic sequential composition of P and Q coincides with the demonic sequential composition, Theorem 3.44b; this was one of the objectives. The assumption $P \circ eqar \circ P^{\cup} \sqsubseteq eqar$ states that the output arity is determined by the input arity. Compare this property for a proc in $\mathcal{A}$ to the property of *detar*, Definition 5.3, for a relation in $\mathcal{B}$. The link is established by Corollary 5.15. Therefore, we extend the definition of *detar* to procs:

Definition 8.6 *Determination of output arity*

$$
\begin{array}{ll}
 & detar.P \\
\equiv & \\
 & P \circ eqar \circ P^{\cup} \sqsubseteq eqar
\end{array}
$$

□

Next, the results of Proposition 8.5 are extended to preservation of causality for arbitrary procs:

Proposition 8.7

$$\begin{array}{ll} & \mathit{causal}.(P \circ Q) \\ \Leftarrow & \\ & \mathit{causal}.P \ \wedge \ \mathit{causal}.Q \ \wedge \ P{>} \sqsupseteq Q{<} \ \wedge \ (\mathit{detar}.P^{\cup} \ \vee \ \mathit{detar}.Q) \end{array}$$

Proof:
Non-emptiness of $P \circ Q$ follows from the type assumptions on P and Q and the non-emptiness of Q. The following fact, expressing preservation of refinement by sequential composition, is used to finish the proof:

$$\begin{array}{ll} & P' \circ Q' \leq P \circ Q \ \wedge \ P'{>} \sqsupseteq Q'{<} \\ \Leftarrow & \\ & P' \leq P \ \wedge \ Q' \leq Q \ \wedge \ P{>} \sqsupseteq Q{<} \end{array}$$

Moreover,

$$\begin{array}{ll} & \mathit{detar}.P' \\ \Leftarrow & \\ & P' \sqsubseteq P \ \wedge \ \mathit{detar}.P \end{array}$$

Then:

$$\begin{array}{ll} & P \circ Q \\ \sqsubseteq & \quad \{ \ \mathit{causal}.P \text{ and } \mathit{causal}.Q \ \} \\ & \sqcup(\, P' \ : \ P' \leq P \wedge \mathit{caus}.P' \ : \ P' \,) \circ \sqcup(\, Q' \ : \ Q' \leq Q \wedge \mathit{caus}.Q' \ : \ Q' \,) \\ = & \quad \{ \text{ cupjunctivity of composition } \} \\ & \sqcup(\, P', Q' \ : \ P' \leq P \wedge Q' \leq Q \wedge \mathit{caus}.P' \wedge \mathit{caus}.Q' \ : \ P' \circ Q' \,) \\ \sqsubseteq & \quad \{ \text{ above facts: Proposition 8.5 } \} \\ & \sqcup(\, P', Q' \ : \ P' \circ Q' \leq P \circ Q \ \wedge \ \mathit{caus}.(P' \circ Q') \ : \ P' \circ Q' \,) \\ \sqsubseteq & \quad \{ \text{ plat calculus } \} \\ & \sqcup(\, R \ : \ R \leq P \circ Q \ \wedge \ \mathit{caus}.R \ : \ R \,) \end{array}$$

□

Again, observe that Proposition 8.7 is a true generalisation of Proposition 8.5. The assumption $\mathit{detar}.P$ is implied by the property $P \sqsubseteq \mathit{eqar}$, which describes that the output arity *is the same* as the input arity. The condition $P{>} \sqsupseteq Q{<}$ is implied by $P{>} = I$. In calculations, all this information will often be captured, together with causality, by (primed) typing: $P \in I' \overset{c}{\rightsquigarrow} I'$.

8.2.3 Parallel composition

Compared to the preservation of causality by sequential composition, the preservation property for parallel composition is more difficult:

Proposition 8.8

$$\mathit{caus}.(P \parallel Q) \ \Leftarrow \ \mathit{caus}.P \ \wedge \ \mathit{caus}.Q$$

Proof:

$$
\begin{array}{ll}
 & eqar \sqcap P \parallel Q \circ {}^{<}t \circ (P \parallel Q)^{\cup} \\
= & \quad \{ \text{ Theorem 3.26b twice; Theorem 3.11a } \} \\
 & eqar \circ I \parallel I \sqcap P \parallel Q \circ {}^{<}t \circ P^{\cup} \parallel Q^{\cup} \\
= & \quad \{ \text{ Theorem 5.13e; Proposition 6.11d } \} \\
 & eqar \parallel eqar \sqcap P \parallel Q \circ {}^{<}t \parallel {}^{<}t \circ P^{\cup} \parallel Q^{\cup} \\
= & \quad \{ \text{ Parallel-parallel fusion 3.12b; capjunctivity of parallel composition } \} \\
 & (eqar \sqcap P \circ {}^{<}t \circ P^{\cup}) \parallel (eqar \sqcap Q \circ {}^{<}t \circ Q^{\cup}) \\
\sqsubseteq & \quad \{ \text{ inertia of } P \text{ and } Q \} \\
 & {}^{<}(\alpha \bullet t) \parallel {}^{<}(\beta \bullet t) \\
\sqsubseteq & \quad \{ \text{ Proposition A.10 } \} \\
 & {}^{<}(\downarrow \bullet \alpha \blacktriangle \beta \bullet t) \parallel {}^{<}(\downarrow \bullet \alpha \blacktriangle \beta \bullet t) \\
\sqsubseteq & \quad \{ \text{ Proposition 6.11e } \} \\
 & {}^{<}(\downarrow \bullet \alpha \blacktriangle \beta \bullet t)
\end{array}
$$

The non-emptiness of $P \parallel Q$ follows from non-emptiness of P and Q and Theorem 3.20c. For the other part, a distributivity property of composition over cap, and capjunctivity of parallel composition is used:

$$
\begin{array}{ll}
 & \sqcap(\, t \ :: \ \top\!\!\top \circ P \parallel Q \circ {}^{<}t \,) \\
= & \quad \{ \ \top\!\!\top \circ I \parallel I = \top\!\!\top \circ \top\!\!\top \parallel \top\!\!\top ; \text{ Proposition 6.11d } \} \\
 & \sqcap(\, t \ :: \ \top\!\!\top \circ \top\!\!\top \parallel \top\!\!\top \circ P \parallel Q \circ {}^{<}t \parallel {}^{<}t \,) \\
= & \quad \{ \text{ Parallel-parallel fusion 3.12b } \} \\
 & \sqcap(\, t \ :: \ \top\!\!\top \circ (\top\!\!\top \circ P \circ {}^{<}t) \parallel (\top\!\!\top \circ Q \circ {}^{<}t) \,) \\
= & \quad \{ \text{ non-trivial distribution over cap } \} \\
 & \top\!\!\top \circ \sqcap(\, t \ :: \ (\top\!\!\top \circ P \circ {}^{<}t) \parallel (\top\!\!\top \circ Q \circ {}^{<}t) \,) \\
= & \quad \{ \text{ capjunctivity of parallel composition } \} \\
 & \top\!\!\top \circ \sqcap(\, t \ :: \ \top\!\!\top \circ P \circ {}^{<}t \,) \parallel \sqcap(\, t \ :: \ \top\!\!\top \circ Q \circ {}^{<}t \,) \\
\sqsubseteq & \quad \{ \ P \text{ and } Q \text{ are closed } \} \\
 & \top\!\!\top \circ (\top\!\!\top \circ P) \parallel (\top\!\!\top \circ Q) \\
= & \quad \{ \ \top\!\!\top \circ \top\!\!\top \parallel \top\!\!\top = \top\!\!\top \circ I \parallel I \ \} \\
 & \top\!\!\top \circ P \parallel Q
\end{array}
$$

□

The implication in Proposition 8.8 is actually an equivalence. Part of this claim follows from Theorem 3.20c and Proposition 7.25.
The previous results are extended to causal procs:

Property 8.9 *Parallel composition*

$$causal.(P \parallel Q) \ \Leftarrow \ causal.P \wedge causal.Q$$

Proof:
In the proof the property $P' \parallel Q' \leq P \parallel Q \;\Leftarrow\; P' \leq P \wedge Q' \leq Q$ is used. For the non-emptiness, Theorem 3.20c suffices. Then:

$$
\begin{array}{ll}
 & P \parallel Q \\
\sqsubseteq & \quad \{\ causal.P \text{ and } causal.Q\ \} \\
 & \sqcup(\, P' \;:\; P' \leq P \wedge caus.P' \;:\; P' \,) \parallel \sqcup(\, Q' \;:\; Q' \leq Q \wedge caus.Q' \;:\; Q' \,) \\
= & \quad \{\ \text{cupjunctivity of parallel composition}\ \} \\
 & \sqcup(\, P', Q' \;:\; P' \leq P \wedge Q' \leq Q \wedge caus.P' \wedge caus.Q' \;:\; P' \parallel Q' \,) \\
\sqsubseteq & \quad \{\ \text{above fact; Proposition 8.8}\ \} \\
 & \sqcup(\, P', Q' \;:\; P' \parallel Q' \leq P \parallel Q \;\wedge\; caus.(P' \parallel Q') \;:\; P' \parallel Q' \,) \\
\sqsubseteq & \quad \{\ \text{plat calculus}\ \} \\
 & \sqcup(\, R \;:\; R \leq P \parallel Q \;\wedge\; caus.R \;:\; R \,)
\end{array}
$$

□

8.2.4 Split

We turn to split, which is also part of the axiomatisation of parallel composition. For the inertia-part, we first prove a small lemma:

Proposition 8.10

$$I \vartriangle I \circ {<}t \circ (I \vartriangle I)^{\cup} \sqsubseteq {<}t$$

Proof:

$$
\begin{array}{ll}
 & I \vartriangle I \circ {<}t \circ (I \vartriangle I)^{\cup} \\
\sqsubseteq & \quad \{\ P \vartriangle Q \circ R \sqsubseteq (P \circ R) \vartriangle (Q \circ R)\ \} \\
 & {<}t \vartriangle {<}t \circ (I \vartriangle I)^{\cup} \\
= & \quad \{\ \text{Parallel-split fusion 3.12a}\ \} \\
 & {<}t \parallel {<}t \circ I \vartriangle I \circ (I \vartriangle I)^{\cup} \\
\sqsubseteq & \quad \{\ I \vartriangle I \text{ is a function: Definition 3.33a}\ \} \\
 & {<}t \parallel {<}t \\
\sqsubseteq & \quad \{\ \text{Proposition 6.11e}\ \} \\
 & {<}t
\end{array}
$$

□

This proposition makes it possible to derive the inertia preservation of $\vartriangle$ from the inertia-preserving properties of parallel composition:

Proposition 8.11

$$caus.(P \vartriangle Q) \;\Leftarrow\; caus.P \;\wedge\; caus.Q \;\wedge\; P \circ Q^{\cup} \neq \perp\!\!\!\perp$$

Proof:
The extra conjunct $P \circ Q^{\cup} \neq \perp\!\!\!\perp$ can be interpreted as the property that P and Q have some source domain elements in common.

The inertia of $P \triangle Q$ follows from the (proof of) inertia of $P \parallel Q$:

$$
\begin{array}{ll}
 & eqar \sqcap P \triangle Q \circ {}^{<}t \circ (P \triangle Q)^{\cup} \\
= & \quad \{ \text{ Parallel-split fusion 3.12a; reverse over composition } \} \\
 & eqar \sqcap P \parallel Q \circ I \triangle I \circ {}^{<}t \circ (I \triangle I)^{\cup} \circ (P \parallel Q)^{\cup} \\
\sqsubseteq & \quad \{ \text{ Proposition 8.10 } \} \\
 & eqar \sqcap P \parallel Q \circ {}^{<}t \circ (P \parallel Q)^{\cup} \\
\sqsubseteq & \quad \{ \text{ inertia of } P \text{ and } Q\text{; proof of inertia in Proposition 8.8 } \} \\
 & {}^{<}(\downarrow \bullet \alpha \blacktriangle \beta \bullet t)
\end{array}
$$

For the non-emptiness we only remark that $R \triangle S \neq \perp\!\!\!\perp \equiv R \circ S^{\cup} \neq \perp\!\!\!\perp$. The preservation of limits by $P \triangle Q$ follows from a calculation which is analogous to the proof of preservation by $P \parallel Q$

□

The previous results are extended to arbitrary causal procs:

Property 8.12 *Split*

$$causal.(P \triangle Q) \;\Leftarrow\; causal.P \;\wedge\; causal.Q \;\wedge\; P \circ Q^{\cup} \neq \perp\!\!\!\perp$$

□

In the proof the property

$$
\begin{array}{ll}
 & P' \triangle Q' \leq P \triangle Q \;\wedge\; P' \circ Q'^{\cup} \neq \perp\!\!\!\perp \\
\Leftarrow & \\
 & P' \leq P \;\wedge\; Q' \leq Q \;\wedge\; P \circ Q^{\cup} \neq \perp\!\!\!\perp
\end{array}
$$

is used. The proof is omitted because it strongly resembles the proof of Property 8.9.

8.2.5 Feedback

This subsection concerns the feedback operator. During the calculations, properties of points and the principles of extensionality and induction are used. However, only the final results are going to be axiomatised, in that way abstracting from the use of extensionality and induction.

Proposition 8.13

$$caus.(P^{\sigma}) \;\Leftarrow\; caus.P \;\wedge\; P \in A' \rightsquigarrow B \parallel A'$$

Proof:
The proof of this proposition requires a preliminary result which is 'expensive' in the sense that the proof is not straightforward and the applicability of the result itself is very restricted:

$$
\begin{array}{ll}
 & \mu P \circ eqar \circ \mu Q \sqsubseteq {}^{<}t_0 \;\Leftarrow\; eqar \sqcap P \circ Q^{\cup} \sqsubseteq {}^{<}t_0 \\
\Leftarrow & \\
 & inert.P \wedge P \sqsubseteq eqar \wedge Q \sqsubseteq eqar \wedge P{>} = Q{>}
\end{array}
$$

Let the archimedean function α be a witness for the inertia of P in:

$$
\begin{array}{ll}
 & \mu P \circ eqar \circ \mu Q \sqsubseteq {<}t_0 \\
= & \quad \{ \text{ Proposition 6.11a } \} \\
 & \mu P \circ eqar \circ \mu Q \sqsubseteq I \sqcup {<}t_0 \\
= & \quad \{ \text{ Propositions 6.14b and 7.6 } \} \\
 & \mu P \circ \sqcup(\, t \ ::\ {<}t \,) \circ \mu Q \ \sqsubseteq\ \sqcap(\, n \ ::\ {<}(\alpha^n \bullet t) \,) \sqcup {<}t_0 \\
= & \quad \{ \text{ cupjunctivity of composition; capjunctivity of cup } \} \\
 & \sqcup(\, t \ ::\ \mu P \circ {<}t \circ \mu Q \,) \sqsubseteq \sqcap(\, n \ ::\ {<}(\alpha^n \bullet t) \sqcup {<}t_0 \,) \\
= & \quad \{ \text{ plat calculus } \} \\
 & \forall(\, t, n \ ::\ \mu P \circ {<}t \circ \mu Q \sqsubseteq {<}(\alpha^n \bullet t) \sqcup {<}t_0 \,) \\
\Leftarrow & \quad \{ \text{ induction } \} \\
 & \forall(\, t \ ::\ \mu P \circ {<}t \circ \mu Q \sqsubseteq {<}(\alpha \bullet t) \sqcup {<}t_0 \,) \\
= & \quad \{ \text{ Proposition 6.13a } \} \\
 & \forall(\, t \ ::\ \mu P \circ {<}t \circ \mu Q \sqsubseteq {<}(\alpha \bullet t) \circ {<}t_0 \,) \\
\Leftarrow & \quad \{ \ \mu P \sqsubseteq P \circ P{>} = P \ \} \\
 & \forall(\, t \ ::\ P \circ {<}t \circ Q{>} \circ Q^{\cup} \sqsubseteq {<}(\alpha \bullet t) \circ {<}t_0 \,) \\
\Leftarrow & \quad \{ \ Q{>} = P{>} \sqsubseteq P^{\cup} \circ P \ \} \\
 & \forall(\, t \ ::\ P \circ {<}t \circ P^{\cup} \circ P \circ Q^{\cup} \sqsubseteq {<}(\alpha \bullet t) \circ {<}t_0 \,) \\
\Leftarrow & \quad \{ \text{ monotonicity } \} \\
 & \forall(\, t \ ::\ P \circ {<}t \circ P^{\cup} \sqsubseteq {<}(\alpha \bullet t) \,) \ \wedge \ P \circ Q^{\cup} \sqsubseteq {<}t_0 \\
= & \quad \{ \ P \sqsubseteq eqar \text{ and } Q \sqsubseteq eqar \ \} \\
 & \forall(\, t \ ::\ eqar \sqcap P \circ {<}t \circ P^{\cup} \ \sqsubseteq\ {<}(\alpha \bullet t) \,) \ \wedge \ eqar \sqcap P \circ Q^{\cup} \ \sqsubseteq\ {<}t_0 \\
= & \quad \{ \ P \text{ is inert } \} \\
 & eqar \sqcap P \circ Q^{\cup} \ \sqsubseteq\ {<}t_0
\end{array}
$$

In the proof, the principle of induction is applied. The step is taken without further explanation to avoid a cumbersome calculation that would not contribute to the main calculation. The result derived above is the instrument to tackle 8.13. First we discuss the inertia part. The proof obligation is that there exists an archimedean function α such that for all moments t:

$$
\begin{array}{ll}
 & eqar \sqcap P^{\sigma} \circ {<}t \circ (P^{\sigma})^{\cup} \ \sqsubseteq\ {<}(\alpha \bullet t) \\
= & \quad \{ \text{ Lemma 7.29; reverse } \} \\
 & eqar \sqcap P^{\sigma} \circ B \circ {<}t \circ B \circ (P^{\sigma})^{\cup} \ \sqsubseteq\ {<}(\alpha \bullet t) \\
= & \quad \{ \text{ Extensionality } \} \\
 & \forall(\, x, y \ :\ x \circ y^{\cup} \sqsubseteq B \circ {<}t \circ B \ :\ eqar \sqcap P^{\sigma} \circ x \circ y^{\cup} \circ (P^{\sigma})^{\cup} \ \sqsubseteq\ {<}(\alpha \bullet t) \,)
\end{array}
$$

The assumptions on P that can be used are $P \in A' \rightsquigarrow B \parallel A'$ and the causality of P. According to Lemma 7.23, we can conclude that for any $z \in B$: $(P \circ z \triangle I){>} = A$ and $P \circ z \triangle I \sqsubseteq eqar$. Moreover, Proposition 7.24 states that $P \circ z \triangle I$ is even inert. This enables us to exploit the preliminary result.
We continue: for all x and y such that $x \circ y^{\cup} \sqsubseteq B \circ {<}t \circ B$:

$$
\begin{array}{ll}
 & eqar \sqcap P^{\sigma} \circ x \circ y^{\cup} \circ (P^{\sigma})^{\cup} \ \sqsubseteq\ {<}(\alpha \bullet t) \\
= & \quad \{ \text{ reverse through composition; Lemma 7.22 } \}
\end{array}
$$

$$
\begin{array}{ll}
 & \mathit{eqar} \sqcap \mu(P \circ x \vartriangle I) \circ \top\!\!\top \circ \mu(P \circ y \vartriangle I) \sqsubseteq {}^{<}(\alpha \bullet t) \\
= & \quad \{\ Q \sqcap A \circ \top\!\!\top \circ B \ =\ A \circ Q \circ B\ \} \\
 & \mu(P \circ x \vartriangle I) \circ \mathit{eqar} \circ \mu(P \circ y \vartriangle I) \sqsubseteq {}^{<}(\alpha \bullet t) \\
\Leftarrow & \quad \{\ \text{above-mentioned facts: preliminary result}\ \} \\
 & \mathit{eqar} \sqcap P \circ x \vartriangle I \circ (P \circ y \vartriangle I)^{\cup} \sqsubseteq {}^{<}(\alpha \bullet t) \\
= & \quad \{\ \text{reverse through composition}\ \} \\
 & \mathit{eqar} \sqcap P \circ x \vartriangle I \circ (y \vartriangle I)^{\cup} \circ P^{\cup} \sqsubseteq {}^{<}(\alpha \bullet t) \\
\Leftarrow & \quad \{\ R \vartriangle I \circ (S \vartriangle I)^{\cup} \sqsubseteq (R \circ S^{\cup}) \parallel I\ \} \\
 & \mathit{eqar} \sqcap P \circ (x \circ y^{\cup}) \parallel I \circ P^{\cup} \sqsubseteq {}^{<}(\alpha \bullet t) \\
\Leftarrow & \quad \{\ \text{close quantification over } x \text{ and } y\text{: } x \circ y^{\cup} \sqsubseteq B \circ {}^{<}t \circ B\ \} \\
 & \mathit{eqar} \sqcap P \circ (B \circ {}^{<}t \circ B) \parallel I \circ P^{\cup} \sqsubseteq {}^{<}(\alpha \bullet t) \\
\Leftarrow & \quad \{\ \text{Propositions 6.11a and 6.11e: } (B \circ {}^{<}t \circ B) \parallel I \sqsubseteq {}^{<}t\ \} \\
 & \mathit{eqar} \sqcap P \circ {}^{<}t \circ P^{\cup} \sqsubseteq {}^{<}(\alpha \bullet t)
\end{array}
$$

which equivales the inertia of P. The property of closedness of P^{σ} is delightfully simple:

$$
\begin{array}{ll}
 & \mathit{closed}.P^{\sigma} \\
= & \quad \{\ \text{Proposition 7.18; assumptions on } P\text{: Proposition 7.30}\ \} \\
 & \mathit{closed}.B \\
\Leftarrow & \quad \{\ \text{Proposition 7.25}\ \} \\
 & \mathit{closed}.(B \parallel A) \\
= & \quad \{\ \text{assumption on } P\text{: Proposition 7.18}\ \} \\
 & \mathit{closed}.P
\end{array}
$$

□

Finally, there is just one property left to be proved:

Property 8.14 *Feedback*

$$\mathit{causal}.(P^{\sigma}) \ \Leftarrow\ P \in A' \overset{c}{\rightsquigarrow} B \parallel A'$$

Proof:
The non-emptiness of P^{σ} is proved by contraposition:

$$
\begin{array}{ll}
 & P^{\sigma} = \bot\!\!\bot \\
= & \quad \{\ P \in A' \overset{c}{\rightsquigarrow} B \parallel A'\text{: Property 7.36 and Theorem 3.39a}\ \} \\
 & B = \bot\!\!\bot \\
\Rightarrow & \quad \{\ \text{parallel composition is } \bot\!\!\bot\text{-strict}\ \} \\
 & B \parallel A = \bot\!\!\bot \\
= & \quad \{\ P \in A \rightsquigarrow B \parallel A\text{: Theorem 3.39a}\ \} \\
 & P = \bot\!\!\bot \\
= & \quad \{\ \mathit{causal}.P\ \} \\
 & \mathit{False}
\end{array}
$$

The proof of causality continues:

$$
\begin{array}{ll}
 & P^{\sigma} \\
\sqsubseteq & \quad \{\ \mathit{causal}.P\text{: Characterisation 7.33; monotonicity of feedback}\ \}
\end{array}
$$

$$
\begin{array}{ll}
 & (\sqcup(\, Q \;:\; Q \leq P \wedge \mathit{caus}.Q \;:\; Q \,))^\sigma \\
= & \qquad \{ \text{ Theorem 3.16 } \} \\
 & \sqcup(\, Q \;:\; Q \leq P \wedge \mathit{caus}.Q \;:\; Q^\sigma \,) \\
\sqsubseteq & \qquad \{ \text{ below: } Q^\sigma \leq P^\sigma \wedge \mathit{caus}.(Q^\sigma) \Leftarrow Q \leq P \wedge \mathit{caus}.Q \ \} \\
 & \sqcup(\, Q \;:\; Q^\sigma \leq P^\sigma \wedge \mathit{caus}.(Q^\sigma) \;:\; Q^\sigma \,) \\
\sqsubseteq & \qquad \{ \text{ plat calculus } \} \\
 & \sqcup(\, R \;:\; R \leq P^\sigma \wedge \mathit{caus}.R \;:\; R \,)
\end{array}
$$

The assumptions of Property 8.14, $P \in A' \overset{c}{\rightsquigarrow} B \parallel A'$, together with the refinement $Q \leq P$ and $\mathit{caus}.Q$, give us:

$$
\begin{array}{ll}
 & Q^\sigma \leq P^\sigma \wedge \mathit{caus}.(Q^\sigma) \\
= & \qquad \{ \text{ Definition 7.31 } \} \\
 & Q^\sigma \sqsubseteq P^\sigma \ \wedge \ (Q^\sigma)^{\triangleright} = (P^\sigma)^{\triangleright} \ \wedge \ \mathit{caus}.(Q^\sigma) \\
\Leftarrow & \qquad \{ \text{ monotonicity feedback; Theorem 3.39a } \} \\
 & Q \sqsubseteq P \ \wedge \ Q^\sigma \in A \rightsquigarrow B \ \wedge \ P^\sigma \in A \rightsquigarrow B \ \wedge \ \mathit{caus}.(Q^\sigma) \\
\Leftarrow & \qquad \{ \text{ Definition 7.31; Proposition 7.30 and Property 7.36 } \} \\
 & Q \leq P \ \wedge \ Q \in A' \rightsquigarrow B \parallel A' \ \wedge \ \mathit{caus}.Q \ \wedge \ P \in A' \overset{c}{\rightsquigarrow} B \parallel A' \\
= & \qquad \{ \ Q \leq P \ \wedge \ P \in A' \rightsquigarrow B \parallel A' \text{: } Q \in A' \rightsquigarrow B \parallel A' \ \} \\
 & Q \leq P \ \wedge \ \mathit{caus}.Q \ \wedge \ P \in A' \overset{c}{\rightsquigarrow} B \parallel A' \\
= & \qquad \{ \text{ assumptions on } P \ \} \\
 & Q \leq P \wedge \mathit{caus}.Q
\end{array}
$$

□

Chapter 9

Weak causality

In Proposition 8.5 the inertia preservation of sequential composition was proved. The witness for inertia of $P \circ Q$ for inert P and Q used in the proof did not depend on the witness for inertia of Q. Of course, one can give the 'symmetric' proof of the proposition and show that inertia of $P \circ Q$ is established by the witness of Q *only*.

9.1 Weak causality defined

The above discussion suggests that plain inertia of both arguments of sequential composition is really overkill. More specifically: the (new) assumption on inertia of Q in Proposition 8.5

$$\forall(\, t \;::\; eqar \sqcap Q \circ {<}t \circ Q^{\cup} \sqsubseteq {<}t \,)$$

is sufficient to get the same result by an even shorter proof. This assumption on Q is dubbed *weak inertia* and defined as follows:

Characterisation 9.1 *Weak inertia*

$$\begin{array}{ll} & winert.P \\ \equiv & \\ & \forall(\, t \;::\; eqar \sqcap P \circ {<}t \circ P^{\cup} \sqsubseteq {<}t \,) \end{array}$$

□

Remember that, in words, inertia expresses the fact that the proc does not react instantaneously, nor does it look into the future. The requirement of 'no instantaneous reaction' is dropped in the definition of weak inertia. This weaker formulation of inertia induces a weaker formulation of causality:

Characterisation 9.2 *Weak causality*

a. $wcaus.P \triangleq winert.P \wedge closed.P$

b. $wcausal.P \triangleq \perp\!\!\!\perp \neq P = \sqcup(\, Q \;:\; Q \leq P \wedge wcaus.Q \;:\; Q \,)$

In combination with typing, the notation '$\overset{w}{\sim}$' is used to denote that the proc is weakly causal in the sense of 9.2b
□

The following section summarises a number of properties concerning the new notion of weak causality.

9.2 Properties

The property of inertia implies weak inertia. Consequently, causality implies weak causality:

Property 9.3 *Weak causality*

$$wcausal.P \Leftarrow causal.P$$

□

The proof of Property 9.3, being straightforward, is omitted. In Section 8.1 it was shown that several basic procs like I and the projections are not causal (assuming the reasonable axiom $\top\!\!\top \neq I$). It will not come as a surprise, though, that these functions are weakly causal.
The weak causality of the identity i and the projections is stated and proved in the following property. For reasons of typing, we show a stronger result for *eqsp*:

Property 9.4 *Weak causality of constants*

a. $wcausal.i$

b. $wcausal.\ll$

c. $wcausal.\gg$

d. $\forall(\, X \;:\; X \in \mathcal{D} \;:\; X \circ eqsp \in I \overset{w}{\sim} I \,)$

Proof:
Because i is an non-empty interface, weak causality follows from $closed.i$. As an example, we show 9.4b. Because $\ll$ is functional, we only have to show that it is *wcaus*; this follows from a result similar to Proposition 7.34a. The proof of 9.4b is shown first:

$$\begin{array}{ll} & winert.\ll \\ = & \quad \{\ \text{Characterisation 9.1}\ \} \\ & \forall(\, t \;::\; eqar \sqcap \ll \circ {}^{<}t \circ \ll^{\cup} \sqsubseteq {}^{<}t \,) \\ \Leftarrow & \quad \{\ \text{plat calculus; Proposition 6.11d}\ \} \\ & \forall(\, t \;::\; \ll \circ {}^{<}t \parallel {}^{<}t \circ \ll^{\cup} \sqsubseteq {}^{<}t \,) \\ = & \quad \{\ \text{Parallel computation 3.32c}\ \} \\ & \forall(\, t \;::\; {}^{<}t \circ \ll \circ I \parallel ({}^{<}t)^{>} \circ \ll^{\cup} \sqsubseteq {}^{<}t \,) \end{array}$$

$$
\begin{array}{ll}
\Leftarrow & \{\ I \parallel ({}^{<}t)^{>} \sqsubseteq I\ \} \\
& \forall(\, t\ ::\ {}^{<}t \circ \ll \circ \ll^{\cup} \sqsubseteq {}^{<}t\,) \\
= & \{\ \ll \text{ is a function: } \ll \circ \ll^{\cup} \sqsubseteq I\ \} \\
& \mathit{True}
\end{array}
$$

Non-emptiness of $\ll$ is shown by contraposition:

$$
\begin{array}{ll}
& \ll = \perp\!\!\!\perp \\
\Rightarrow & \{\ \text{sequential composition is } \perp\!\!\!\perp\text{-strict}\ \} \\
& \ll \circ\ \top\!\!\top \vartriangle \top\!\!\top\ =\ \perp\!\!\!\perp \\
= & \{\ \text{Split computation 3.13a}\ \} \\
& \top\!\!\top \sqcap \top\!\!\top \circ \top\!\!\top\ =\ \perp\!\!\!\perp \\
= & \{\ \text{Theorem 3.6; } \top\!\!\top \text{ is unit of cap}\ \} \\
& \top\!\!\top = \perp\!\!\!\perp \\
= & \{\ \text{Theorem 3.20b}\ \} \\
& \mathit{False}
\end{array}
$$

And, finally, $\ll^{>} = I \parallel I$ is closed under limits:

$$
\begin{array}{ll}
& \sqcap(\, t\ ::\ \top\!\!\top \circ I \parallel I \circ {}^{<}t\,) \\
= & \{\ \text{Proposition 6.11d}\ \} \\
& \sqcap(\, t\ ::\ \top\!\!\top \circ {}^{<}t \parallel {}^{<}t\,) \\
= & \{\ \text{Theorem 3.30c; domains distribute over parallel composition}\ \} \\
& \sqcap(\, t\ ::\ \top\!\!\top \circ ({}^{<}t)^{>} \parallel ({}^{<}t)^{>}\,) \\
\sqsubseteq & \{\ \text{Theorem 3.30d}\ \} \\
& \sqcap(\, t\ ::\ \top\!\!\top \circ I \parallel I\,) \\
= & \{\ \text{plat calculus}\ \} \\
& \top\!\!\top \circ I \parallel I
\end{array}
$$

We continue with 9.4d. Let X be an element of $\mathcal{D}$, Definition 7.7. The proc $X \circ \mathit{eqsp}$ is too non-deterministic to be causal in the sense of 7.20 or 9.2a. Furthermore, because the proc connects two supports and consequently exhibits instantaneous response, it is not causal in the sense of Characterisation 7.33 either. We will show that $X \circ \mathit{eqsp}$ is weakly causal according to Characterisation 9.2b.
The proc $X \circ \mathit{eqsp}$ has to be decomposed into polyfunctions which return, given some chronicle g, a chronicle f with arity X and the same support as g: $\mathit{sp}.f = \mathit{sp}.g$. For that purpose we define, just like the set of chronicles $\mathcal{C}$, Characterisation 4.13, a set $\mathcal{E}$ containing functions with their support equal to $\mathcal{T}$. For the basis, the set bs is defined as follows:

$$\mathit{bs} \triangleq \{\, x \mid x \in \iota \leftrightarrowtail \mathcal{T}\ \wedge\ \mathit{sp}.x = \mathcal{T}\,\}$$

For all predicates Pr on relations, the following equivalence holds:

$$
\begin{array}{ll}
& \forall(\, x\ :\ x \in \mathcal{E}\ :\ Pr.x\,) \\
\equiv & \\
& \forall(\, x\ :\ x \in \mathit{bs}\ :\ Pr.x\,) \\
& \wedge\ \forall(\, y, z\ :\ y \in \mathcal{E} \wedge z \in \mathcal{E} \wedge Pr.y \wedge Pr.z\ :\ Pr.(y \blacktriangle z)\,)
\end{array}
$$

We let the identifier $\mathbf{f}$ range over $\mathcal{E}$ with arity X. A chronicle can be constructed by restricting the support of $\mathbf{f}$ to a well-ordered set, and brushing away the remaining messages. To obtain this behaviour, the function $[\mathbf{f}]$ is defined point-wise for all f and g:

$$\begin{array}{l} \quad f\,\langle[\mathbf{f}]\rangle\,g \\ \equiv \\ \quad f \;=\; \mathbf{f} \bullet sp.g \cup wipe \bullet \mathbf{f} \bullet \widetilde{sp.g} \end{array}$$

where $\widetilde{sp.g} = \neg sp.g \cap \mathcal{T}$. It follows from this definition that the proc $[\mathbf{f}]$ is total. The function $\mathbf{f}$ fixes the output arity of f to X. It follows that $f\,\langle[\mathbf{f}]\rangle\,g$ implies $sp.f = sp.g$. On the other hand, a function $\mathbf{f} \in \mathcal{E}$ such that $f\,\langle[\mathbf{f}]\rangle\,g$ can be constructed for *any* pair $f \in X$ and g with $sp.f = sp.g$: replace all no-messages in f by real messages. This motivates the claim that $X \circ eqsp$ can be written as the union of all $[\mathbf{f}]$-structures: $X \circ eqsp = \sqcup(\,\mathbf{f} \;::\; [\mathbf{f}]\,)$. The $[\mathbf{f}]$-structures are refinements of $X \circ eqsp$, total and weakly causal in the sense of 9.2a. Consequently, according to Characterisation 9.2b, $X \circ eqsp$ is total and weakly causal.
Now that weak causality is proved for $X \circ eqsp$, we shall have no further use for the set $\mathcal{E}$ or the formulation $\sqcup(\,\mathbf{f} \;::\; [\mathbf{f}]\,)$
□

The kind of strengthened result such as the one proved in Property 9.4d occurs often when the main proc (in this case *eqsp*) absorbs *eqsp*: because of the absorption, all information about the arities is lost. We will encounter similar properties in Section 12.3 and Chapter 14.
With the notion of weak causality, Proposition 8.7 can be strengthened by weakening the assumptions on the arguments:

Property 9.5 *Sequential composition*

$$\begin{array}{l} \quad causal.(P \circ Q) \\ \Leftarrow \\ \quad ((causal.P \wedge wcausal.Q) \vee (wcausal.P \wedge causal.Q)) \;\wedge\; P^{>} \sqsupseteq Q^{<} \\ \qquad \wedge\; (detar.P^{\cup} \vee detar.Q) \end{array}$$

□

The proof of this property is almost an exact copy of the proofs of Proposition 8.5 and Proposition 8.7. An easy exercise is the preservation of weak causality by several compositions. Let $\mathcal{S}$ be a non-empty set of procs:

Property 9.6 *Preservation of weak causality*

a. $\quad wcausal.(\sqcup\mathcal{S}) \;\Leftarrow\; \forall(\,P \;:\; P \in \mathcal{S} \;:\; P \in I \overset{w}{\rightsquigarrow} A\,)$

b.
$$\begin{array}{l} \quad wcausal.(\sqcup\mathcal{S}) \\ \Leftarrow \\ \quad \forall(\,P,Q \;:\; P,Q \in \mathcal{S} \;:\; wcausal.P \;\wedge\; (P \neq Q \;\Rightarrow\; P \circ eqar \circ Q^{\cup} = \perp\!\!\!\perp)\,) \end{array}$$

c. $wcausal.(P \circ Q)$

$\Leftarrow$

$wcausal.P \ \wedge \ wcausal.Q \ \wedge \ P^{>} \sqsupseteq Q^{<} \ \wedge \ (detar.P^{\cup} \vee \ detar.Q)$

d. $wcausal.(P \parallel Q) \ \Leftarrow \ wcausal.P \wedge wcausal.Q$

e. $wcausal.(P \triangle Q) \ \Leftarrow \ wcausal.P \ \wedge \ wcausal.Q \ \wedge \ P \circ Q^{\cup} \neq \perp\!\!\!\perp$

□

Weak causality of the elements in $\mathcal{D}$ is a consequence of Properties 9.4a and 9.6d. This in turn implies weak causality of I by Lemma 7.8a and Property 9.6b. It follows from Properties 9.4d and 9.6a that the proc *eqsp* is weakly causal:

Theorem 9.7 *Weak causality*

a. $\forall(\ X \ : \ X \in \mathcal{D} \ : \ wcausal.X\ )$

b. $wcausal.I$

c. $wcausal.eqsp$

□

Weak causality of P^{σ} for weakly causal P is highly unlikely, because closedness of P^{σ} depends on (plain) inertia of P. It is exactly this part that is dropped in the definition of weak causality. This concludes the theoretical discussion on causality and many preservation properties.

Summarising the results

It has been shown that the class of causal processes is a very important one: a feedback loop preserves polyfunctionality and totality of its argument proc if it is causal. Furthermore, (weak) causality itself is preserved (under certain reasonable conditions) by the composition constructions of the relational algebra.
For completeness, we record all the important results derived in the previous chapters in a new axiom. The notion of *caus*, Characterisation 7.20, only served as an auxiliary notion to get the final one: *causal*, Characterisation 7.33. Therefore, *caus* will not occur in the list below.

Axiom 9.8
First, the constants of the relational algebra:

a. $\neg(wcausal.\perp\!\!\!\perp)$

b. $causal.\top\!\!\!\top$

c. $wcausal.\ll$

d. $wcausal.\gg$

Second, some new primitives of the calculus:

e. $wcausal.i$

f. $causal.eqar$

g. $\forall(\, X \ :\ X \in \mathcal{D} \ :\ X \circ eqsp \in I \overset{w}{\rightsquigarrow} I \,)$

Third, the preservation rules of causality for several composition constructions. For non-empty set $\mathcal{S}$ of procs:

h. $causal.(\sqcup\mathcal{S}) \Leftarrow \forall(\, P \ :\ P \in \mathcal{S} \ :\ P \in I \overset{c}{\rightsquigarrow} A \,)$

i. $causal.(\sqcup\mathcal{S})$
$\Leftarrow$
$\forall(\, P,Q \ :\ P,Q \in \mathcal{S} \ :\ causal.P \ \wedge\ (P \neq Q \ \Rightarrow\ P \circ eqar \circ Q^{\cup} = \perp\!\!\!\perp)\,)$

j. $causal.(P \circ Q)$
$\Leftarrow$
$((causal.P \wedge wcausal.Q) \vee (wcausal.P \wedge causal.Q)) \ \wedge \ P^{>} \sqsupseteq Q^{<}$
$\wedge \ (detar.P^{\cup} \vee detar.Q)$

k. $causal.(P \parallel Q) \ \Leftarrow \ causal.P \wedge causal.Q$

l. $causal.(P \triangle Q) \ \Leftarrow \ causal.P \ \wedge \ causal.Q \ \wedge \ P \circ Q^{\cup} \neq \perp\!\!\!\perp$

m. $causal.P^{\sigma} \ \Leftarrow \ P \ \in \ A' \ \overset{c}{\sim\!\!\!\rightarrow} \ B \parallel A'$

Fourth, the connection between causality and weak causality:

n. $wcausal.P \Leftarrow causal.P$

Fifth, the preservation rules of weak causality for several composition constructions. For non-empty set $\mathcal{S}$ of procs:

o. $wcausal.(\sqcup\mathcal{S}) \ \Leftarrow \ \forall(\, P \ : \ P \in \mathcal{S} \ : \ P \ \in I \ \overset{w}{\sim\!\!\!\rightarrow} \ A \,)$

p. $wcausal.(\sqcup\mathcal{S})$
$\Leftarrow$
$\forall(\, P,Q \ : \ P,Q \in \mathcal{S} \ : \ wcausal.P \ \wedge \ (P \neq Q \ \Rightarrow \ P \circ eqar \circ Q^{\cup} = \perp\!\!\!\perp)\,)$

q. $wcausal.(P \circ Q)$
$\Leftarrow$
$wcausal.P \ \wedge \ wcausal.Q \ \wedge \ P^{>} \sqsupseteq Q^{<} \ \wedge \ (detar.P^{\cup} \vee detar.Q)$

r. $wcausal.(P \parallel Q) \ \Leftarrow \ wcausal.P \wedge wcausal.Q$

s. $wcausal.(P \triangle Q) \ \Leftarrow \ wcausal.P \ \wedge \ wcausal.Q \ \wedge \ P \circ Q^{\cup} \neq \perp\!\!\!\perp$

And finally, the properties it was all about:

t. F^{σ} is a polyfunction $\Leftarrow$ $causal.F$

u. $P^{\sigma} \in A \rightsquigarrow B \ \Leftarrow \ P \ \in \ A' \ \overset{c}{\sim\!\!\!\rightarrow} \ B \parallel A'$

□

Compare these results with the wishes we started with in Chapter 7. Axiom 9.8a is included to be able to infer that (weakly) causal procs are non-empty.

Part IV

Basic procs

Chapter 10

Possible delay, Delay, Prefix

In this chapter the procs *pdly*, *dly* and *pref* are introduced. Those procs make it possible to compare chronicles in several ways. Important properties such as commutation with postcompose-structures and causality are shown to be correct.
It should be noted that the informal interpretation of the procs as given in this chapter and subsequent chapters depends on the assumption $\mathcal{T}$ isomorphic to $\mathbf{R}$. However, almost all results do not depend on this isomorphism. The few that do will be tagged.

10.1 Possible delay

One of the most important procs is a proc expressing the fact that the output chronicle is a delayed version of the input chronicle. This behaviour is exhibited by, for example, a (not too short) wire. If a wire is allowed to have length zero, the proc might respond instantaneously. This behaviour is exhibited by the proc *pdly*. In the next section a real delay *dly* will be considered.
To model a possibly delaying proc, we first define the predicate *pd*:

Characterisation 10.1 *pd*

$$\begin{array}{ll} & pd.\sigma \\ \equiv & \\ & \sigma \subseteq \leqslant \wedge\ oiso.\sigma \end{array}$$

□

By convention, the identifiers σ and σ' range over the functions satisfying *pd*. The property of $pd.\sigma$ describes that σ is a lessening order-isomorphism on $\mathcal{T}$. The identity $\mathcal{T}$ satisfies the condition *pd*. Moreover, *pd* is preserved by composition:

Proposition 10.2

a. $pd.\mathcal{T}$

b. $\exists(\, \sigma, \sigma' \ ::\ \sigma \bullet \sigma' = r\,) \ \equiv\ pd.r$

□

Observe that 10.2b expresses more than just preservation by composition: the implication $\Leftarrow$ follows straightforwardly from 10.2a and is therefore for free. The proc $^\circ\sigma$ has the following interpretation in the model: $f \langle ^\circ\sigma \rangle g \;\equiv\; f = g \bullet \sigma$. Or, in words: f is a 'σ-delayed' version of g (because σ is lessening). The proc $^\circ\sigma$ has some useful properties:

Proposition 10.3
Let ς be an order-isomorphism

a. $\quad {}^\circ\varsigma \circ {}^<(\varsigma \bullet t) = {}^<t \circ {}^\circ\varsigma$

b. $\quad {}^\circ\varsigma \circ {}^\geqslant(\varsigma \bullet t) = {}^\geqslant t \circ {}^\circ\varsigma$

c. $\quad {}^\circ\varsigma \circ r^\circ = r^\circ \circ {}^\circ\varsigma$

Proof:
First:

$$
\begin{array}{ll}
 & {}^\circ\varsigma \circ {}^<(\varsigma \bullet t) = {}^<t \circ {}^\circ\varsigma \\
= & \quad \{ \text{ Characterisation 6.10a } \} \\
 & {}^\circ\varsigma \circ {}^\circ((< \bullet \varsigma \bullet t)_\prec) = {}^\circ((< \bullet t)_\prec) \circ {}^\circ\varsigma \\
= & \quad \{ \text{ } \varsigma \text{ is an isomorphism: compositionality of precompose } \} \\
 & {}^\circ((< \bullet \varsigma \bullet t)_\prec \bullet \varsigma) = {}^\circ(\varsigma \bullet (< \bullet t)_\prec) \\
\Leftarrow & \quad \{ \text{ Leibniz } \} \\
 & (< \bullet \varsigma \bullet t)_\prec \bullet \varsigma = \varsigma \bullet (< \bullet t)_\prec \\
= & \quad \{ \text{ Definition 3.29 } \} \\
 & (< \bullet \varsigma \bullet t \bullet \top \cap \mathcal{I}) \bullet \varsigma \;=\; \varsigma \bullet (< \bullet t \bullet \top \cap \mathcal{I}) \\
= & \quad \{ \text{ Theorem 3.23b; } \varsigma \text{ is an injection: Theorem 3.35a } \} \\
 & < \bullet \varsigma \bullet t \bullet \top \cap \varsigma \;=\; \varsigma \bullet < \bullet t \bullet \top \cap \varsigma \\
\Leftarrow & \quad \{ \text{ Leibniz } \} \\
 & < \bullet \varsigma = \varsigma \bullet < \\
= & \quad \{ \text{ } \varsigma \text{ is monotonic } \} \\
 & True
\end{array}
$$

In the proof of the third statement we use the fact

$$
\begin{array}{ll}
 & a \text{ is well-ordered} \\
\equiv & \\
 & \forall(\, \varsigma' \;:\; oiso.\varsigma' \;:\; (a \bullet \varsigma')_\succ \text{ is well-ordered })
\end{array}
$$

This guarantees that $h \bullet \varsigma$ is a chronicle whenever h is.

$$
\begin{array}{ll}
 & {}^\circ\varsigma \circ r^\circ = r^\circ \circ {}^\circ\varsigma \\
= & \quad \{ \text{ interpretation in the model } \} \\
 & \forall(\, f, g \;::\; \exists(\, h \;::\; f \langle ^\circ\varsigma \rangle h \wedge h \langle r^\circ \rangle g \,) \;\equiv\; \exists(\, h' \;::\; f \langle r^\circ \rangle h' \wedge h' \langle ^\circ\varsigma \rangle g \,) \,) \\
= & \quad \{ \text{ Definition 5.1; } \varsigma \neq \bot \text{: Characterisation 6.2 } \} \\
 & \forall(\, f, g \;::\; \exists(\, h \;::\; f \supseteq h \bullet \varsigma \;\wedge\; h \subseteq r \bullet g \,) \;\equiv\; \exists(\, h' \;::\; f \subseteq r \bullet h' \;\wedge\; h' \supseteq g \bullet \varsigma \,) \,) \\
= & \quad \{ \text{ } \varsigma \text{ is an isomorphism: relational calculus } \} \\
 & \forall(\, f, g \;::\; \exists(\, h \;::\; f \bullet \varsigma^{-1} = h \;\wedge\; h \subseteq r \bullet g \,) \equiv \exists(\, h' \;::\; f \subseteq r \bullet h' \;\wedge\; h' = g \bullet \varsigma \,) \,)
\end{array}
$$

$$
\begin{array}{ll}
= & \{\ h \text{ and } h' \text{ are chronicles: predicate calculus } \} \\
 & \forall(\ f, g\ ::\ f \bullet \varsigma^{-1} \subseteq r \bullet g\ \equiv\ f \subseteq r \bullet g \bullet \varsigma\ ) \\
= & \{\ \varsigma \text{ is a total function: relational calculus } \} \\
 & \forall(\ f, g\ ::\ f \subseteq r \bullet g \bullet \varsigma\ \equiv\ f \subseteq r \bullet g \bullet \varsigma\ ) \\
= & \{\ \text{reflexivity } \} \\
 & True
\end{array}
$$

□

Now, *pdly* is the union over all such $^\circ\sigma$-structures, modelling the property that there might be some (arbitrary but finite) delay:

Characterisation 10.4 *Possible delay*

$$pdly \triangleq \sqcup(\ \sigma\ ::\ ^\circ\sigma\ )$$

□

The use of the word 'possible' stems from the property $\sigma \subseteq \leqslant$ of σ: since $\leqslant$ is reflexive, it is possible that σ is also reflexive (on part of its domain). This can result in (momentarily) instantaneous response of $^\circ\sigma$.
Propositions 10.2 and 10.3 give rise to corresponding properties about *pdly*. This paves the way for an axiomatisation of *pdly*. One of the main properties of *pdly* is that it commutes with postcompose structures:

Property 10.5 *Possible delay*

a. $I \sqsubseteq pdly$

b. $pdly \circ pdly = pdly$

c. $pdly \circ r^\circ = r^\circ \circ pdly$

d. $pdly \circ I \parallel I\ \sqsubseteq\ pdly \parallel pdly$

Proof:
Only the second and the fourth property are handled. The inclusion $pdly \circ pdly \sqsupseteq pdly$ of 10.5b follows from reflexivity of *pdly*, 10.5a.

$$
\begin{array}{ll}
 & pdly \circ pdly \\
= & \{\ \text{Characterisation 10.4 } \} \\
 & \sqcup(\ \sigma\ ::\ ^\circ\sigma\ ) \circ \sqcup(\ \sigma'\ ::\ ^\circ\sigma'\ ) \\
= & \{\ \text{cupjunctivity } \} \\
 & \sqcup(\ \sigma, \sigma'\ ::\ ^\circ\sigma \circ {}^\circ\sigma'\ ) \\
\sqsubseteq & \{\ \text{Proposition 6.3d } \} \\
 & \sqcup(\ \sigma, \sigma'\ ::\ ^\circ(\sigma' \bullet \sigma)\ ) \\
= & \{\ \text{introducing a dummy } \} \\
 & \sqcup(\ r, \sigma, \sigma'\ :\ \sigma' \bullet \sigma = r\ :\ ^\circ r\ ) \\
= & \{\ \text{generalised range disjunction } \}
\end{array}
$$

$$
\begin{array}{ll}
 & \sqcup(\, r \ :\ \exists(\, \sigma, \sigma' \ ::\ \sigma' \bullet \sigma = r \,) \ :\ {}^{\circ}r \,) \\
= & \quad \{ \text{ Proposition 10.2b } \} \\
 & \sqcup(\, r \ :\ pd.r \ :\ {}^{\circ}r \,) \\
= & \quad \{ \text{ Characterisation 10.4 } \} \\
 & pdly
\end{array}
$$

The inclusion that occurs in the above proof is an equality: because σ and σ' are order-isomorphisms, precompose preserves composition. In the proof of the fourth property, we use the fact that the projections in the algebra of procs are just the lifted projections from the algebra of chronicles: $\ll = \ll^{\circ}$ and $\gg = \gg^{\circ}$, Axioms 6.15g and 6.15h. This enables us to exploit Property 10.5c:

$$
\begin{array}{ll}
 & pdly \parallel pdly \\
= & \quad \{ \text{ Definition 3.10 } \} \\
 & \ll^{\cup} \circ pdly \circ \ll \sqcap \gg^{\cup} \circ pdly \circ \gg \\
= & \quad \{ \text{ Axioms 6.15g and 6.15f: } \ll^{\cup} = (\ll^{-1})^{\circ} \text{; Property 10.5c } \} \\
 & pdly \circ \ll^{\cup} \circ \ll \sqcap pdly \circ \gg^{\cup} \circ \gg \\
\sqsupseteq & \quad \{ \text{ monotonicity } \} \\
 & pdly \ \circ \ (\ll^{\cup} \circ \ll \sqcap \gg^{\cup} \circ \gg) \\
= & \quad \{ \text{ Definition 3.10 } \} \\
 & pdly \circ I \parallel I
\end{array}
$$

□

The proc *pdly* represents a one-to-one correspondence between the input and the output chronicle in the sense that each message from one chronicle is related to exactly one message from the other chronicle. The proc $pdly^{\cup} \circ pdly$ abstracts from the timing.

10.2 Delay

This section discusses the delaying proc *dly*. Because several of its properties are similar to corresponding properties of *pdly*, proofs are shortened. To model a delaying proc, the function *dl* is defined first:

Characterisation 10.6 *dl*

$$
\begin{array}{ll}
 & dl.\tau \\
\equiv & \\
 & \tau \subseteq < \ \wedge \ oiso.\tau
\end{array}
$$

□

We let the identifiers τ and τ' range over the functions satisfying *dl*. Compared to Characterisation 10.1, we dropped the part that τ is allowed to be reflexive (on part of its domain). This is expressed most clearly by the fact that $\mathcal{T}$ is *not dl*. We do, however, have that *dl* implies *pd*.

The dual of Proposition 10.2 becomes:

Proposition 10.7

a. $dl.\tau \Rightarrow pd.\tau$

b. $\exists(\, \tau, \tau' \;::\; \tau \bullet \tau' = r \,) \;\Rightarrow\; dl.r$

c. $\exists(\, \tau, \sigma \;::\; \tau \bullet \sigma = r \,) \;\equiv\; dl.r$

□

To get something similar to Proposition 10.2b, we need a property which states that a *dl*-function can be decomposed into two other *dl*-functions. It is clear that this is a false statement in **Z** which is one of the models for $\mathcal{T}$. It is, however, a true statement in **R**. In this set one can show that a *dl*-function τ can be decomposed:

$$\exists(\, \tau', \tau'' \;::\; \tau' \bullet \tau'' = \tau \,)$$

This is a non-trivial result for which Bolzano's Theorem is needed, see Apostol [Apo67]. Recall that a closed infinite chain $(\mathcal{S}, \preceq)$ has all limits of bounded increasing sequences, and that the strict relation $\prec$ is total and surjective on $\mathcal{S}$. We call a closed infinite chain $(\mathcal{S}, \preceq)$ *continuous* if the chain is isomorphic to the chain of reals **R**.
What we can do next is impose the assumption of continuity on $\mathcal{T}$. But then we restrict the applicability of the calculus significantly because all discrete models are cancelled; that is not the objective. What we will do is the following: we will use the continuity of $\mathcal{T}$ whenever it is needed, and tag (all results depending on) the continuity assumption with the symbol • in the left margin.

Assumption 10.8 *Continuous time domain* $\mathcal{T}$

• The closed infinite chain $(\mathcal{T}, \leqslant)$ is isomorphic to the chain of reals **R**

□

We continue with the definition of the proc *dly*. The proc *dly* is the union over all $^\circ\tau$-structures, modelling the property that there is some (arbitrary but finite) delay:

Characterisation 10.9 *Delay*

$$dly \triangleq \sqcup(\, \tau \;::\; {}^\circ\tau \,)$$

□

A shorter definition for *dl* would have been $dl.\tau \equiv arch.\tau^{-1}$, see Characterisation 7.3. The equivalence is no coincidence: it is what makes it possible to show the *causality of dly*. This is exactly the intuitive idea we had about causal procs: every physical machine has some *delay* in its response to input (Section 7.1).
Proposition 10.7 and Proposition 10.3 result in a set of properties for *dly* which are similar to those stated in Property 10.5 for *pdly*. In contrast to *pdly*, the proc *dly* is *not* reflexive.

Observe the tagged Property 10.10b:

Property 10.10 *Delay*

a. $dly \sqsubseteq pdly$

• b. $dly \circ dly = dly$

c. $dly \circ r^\circ = r^\circ \circ dly$

d. $dly \circ I \parallel I \sqsubseteq dly \parallel dly$

e. $dly \circ pdly = dly = pdly \circ dly$

Proof:
To prove 10.10b, in particular the inclusion $\sqsupseteq$, the assumption of continuity is needed. Moreover, the equality ${}^\circ\tau \circ {}^\circ\tau' = {}^\circ(\tau' \bullet \tau)$ appears. It is valid because τ (or τ') is an order-isomorphism:

$$
\begin{array}{ll}
 & dly \circ dly \\
= & \quad \{ \text{ Characterisation 10.9 } \} \\
 & \sqcup(\, \tau \ ::\ {}^\circ\tau \,) \circ \sqcup(\, \tau' \ ::\ {}^\circ\tau' \,) \\
= & \quad \{ \text{ cupjunctivity } \} \\
 & \sqcup(\, \tau, \tau' \ ::\ {}^\circ\tau \circ {}^\circ\tau' \,) \\
= & \quad \{ \ \tau \text{ and } \tau' \text{ are isomorphisms: compositionality of precompose } \} \\
 & \sqcup(\, \tau, \tau' \ ::\ {}^\circ(\tau' \bullet \tau) \,) \\
= & \quad \{ \text{ introducing a dummy } \} \\
 & \sqcup(\, r, \tau, \tau' \ :\ \tau' \bullet \tau = r \ :\ {}^\circ r \,) \\
= & \quad \{ \text{ generalised range disjunction } \} \\
 & \sqcup(\, r \ :\ \exists(\, \tau, \tau' \ ::\ \tau' \bullet \tau = r \,) \ :\ {}^\circ r \,) \\
= & \quad \{ \text{ Assumption 10.8 } \} \\
 & \sqcup(\, r \ :\ dl.r \ :\ {}^\circ r \,) \\
= & \quad \{ \text{ Characterisation 10.9 } \} \\
 & dly
\end{array}
$$

□

Just like the proc *pdly*, *dly* also represents a one-to-one correspondence between the input and the output chronicle. This time in the sense that each message from the input chronicle is related to exactly one message occurring *later* on the output chronicle.

10.3 Prefix

As pointed out at the end of Section 10.1, *pdly* represents a one-to-one correspondence. It may be desirable not to have this strong connection. To express that one chronicle is a

prefix of another chronicle, the proc *pref* is defined. First, we define the more primitive proc *cut*.

10.3.1 Cut

This subsection characterises the proc *cut*. The intention is to use this proc as a building block for another proc, *pref*, which will be defined in the next subsection. All properties of the proc *pref* lean heavily on corresponding properties of *cut*.

Characterisation 10.11 *Cut*

$$cut \triangleq \sqcup(\, t \;::\; {}^{<}t \sqcap {}^{\geqslant}t \circ wipe^{\circ}\,)$$

□

This proc chops its input chronicle at some moment t. All messages which occurred before moment t appear on the output chronicle; all messages which occurred on or after moment t are wiped, that is, no-messages (of the correct type) are produced instead. This is motivated by the interpretation of *cut* in the model:

$$\begin{array}{ll} & f\,\langle cut\rangle\, g \\ \equiv & \\ & \exists(\, t \;::\; f \;=\; g \bullet (<\bullet t)^{\prec} \cup\; wipe \bullet g \bullet (\geqslant \bullet t)^{\prec}\,) \end{array}$$

From this interpretation, one expects that delaying the input chronicle and then cutting at some moment is exactly the same as first cutting the input chronicle on an earlier moment and then delaying the output. It is concisely formulated as $cut \circ pdly = pdly \circ cut$. This proposition is proved as follows:

Proposition 10.12
Let ς be an order-isomorphism.

a. $({}^{<}t \sqcap {}^{\geqslant}t \circ wipe^{\circ}) \;\circ\; {}^{\circ}\varsigma \;=\; {}^{\circ}\varsigma \;\circ\; ({}^{<}(\varsigma \bullet t) \sqcap {}^{\geqslant}(\varsigma \bullet t) \circ wipe^{\circ})$

b. $cut \circ {}^{\circ}\varsigma = {}^{\circ}\varsigma \circ cut$

c. $cut \circ pdly = pdly \circ cut$

d. $cut \circ dly = dly \circ cut$

Proof:

$$\begin{array}{ll} & ({}^{<}t \sqcap {}^{\geqslant}t \circ wipe^{\circ}) \;\circ\; {}^{\circ}\varsigma \\ = & \qquad \{\ \varsigma \text{ is total: Proposition 6.4a and Theorem 3.35a}\ \} \\ & {}^{<}t \circ {}^{\circ}\varsigma \sqcap {}^{\geqslant}t \circ wipe^{\circ} \circ {}^{\circ}\varsigma \\ = & \qquad \{\ \text{Proposition 10.3c}\ \} \\ & {}^{<}t \circ {}^{\circ}\varsigma \sqcap {}^{\geqslant}t \circ {}^{\circ}\varsigma \circ wipe^{\circ} \\ = & \qquad \{\ \text{Propositions 10.3a and 10.3b}\ \} \\ & {}^{\circ}\varsigma \circ {}^{<}(\varsigma \bullet t) \sqcap {}^{\circ}\varsigma \circ {}^{\geqslant}(\varsigma \bullet t) \circ wipe^{\circ} \end{array}$$

$$
\begin{array}{ll}
= & \{\ \varsigma \text{ is surjective: Proposition 6.4b and Theorem 3.35a}\ \} \\
& {}^{\circ}\varsigma \ \circ\ (\ {}^{<}(\varsigma \bullet t) \sqcap {}^{\geqslant}(\varsigma \bullet t) \circ \mathit{wipe}^{\circ})
\end{array}
$$

Proposition 10.12b follows straightforwardly from 10.12a:

$$
\begin{array}{ll}
& \mathit{cut} \circ {}^{\circ}\varsigma \\
= & \quad \{\ \text{Characterisation 10.11; cupjunctivity}\ \} \\
& \sqcup(\ t\ ::\ ({}^{<}t \sqcap {}^{\geqslant}t \circ \mathit{wipe}^{\circ})\ \circ\ {}^{\circ}\varsigma\ ) \\
= & \quad \{\ \text{Proposition 10.12a}\ \} \\
& \sqcup(\ t\ ::\ {}^{\circ}\varsigma\ \circ\ ({}^{<}(\varsigma \bullet t) \sqcap {}^{\geqslant}(\varsigma \bullet t) \circ \mathit{wipe}^{\circ})\ ) \\
= & \quad \{\ \text{cupjunctivity}\ \} \\
& {}^{\circ}\varsigma \circ \sqcup(\ t\ ::\ {}^{<}(\varsigma \bullet t) \sqcap {}^{\geqslant}(\varsigma \bullet t) \circ \mathit{wipe}^{\circ}\ ) \\
= & \quad \{\ \varsigma \text{ is an isomorphism on } \mathcal{T}\text{: dummy change}\ \} \\
& {}^{\circ}\varsigma \circ \sqcup(\ t'\ ::\ {}^{<}t' \sqcap {}^{\geqslant}t' \circ \mathit{wipe}^{\circ}\ ) \\
= & \quad \{\ \text{Characterisation 10.11}\ \} \\
& {}^{\circ}\varsigma \circ \mathit{cut}
\end{array}
$$

The third and fourth statement directly follow from the second one and the definitions of *pdly* and *dly*, Characterisations 10.4 and 10.9
□

More difficult is the idempotency of *cut*. The details of the proof are omitted. For 10.13a we only remark that a case analysis is made: $t \leqslant t'$ or $t \geqslant t'$, in order to use Proposition 6.12. Performing the proof in the model (at the level of chronicles) is less difficult.

Proposition 10.13

a. $({}^{<}t \sqcap {}^{\geqslant}t \circ \mathit{wipe}^{\circ}) \circ ({}^{<}t' \sqcap {}^{\geqslant}t' \circ \mathit{wipe}^{\circ}) \ =\ ({}^{<}t \sqcup {}^{<}t') \ \sqcap\ ({}^{\geqslant}t \sqcap {}^{\geqslant}t') \circ \mathit{wipe}^{\circ}$

b. $\mathit{cut} \circ \mathit{cut} = \mathit{cut}$

Proof:
For 10.13b, we use the instrument of diagonalisation:

$$
\begin{array}{ll}
& \mathit{cut} \circ \mathit{cut} \\
= & \quad \{\ \text{Characterisation 10.11}\ \} \\
& \sqcup(\ t\ ::\ {}^{<}t \sqcap {}^{\geqslant}t \circ \mathit{wipe}^{\circ}\ ) \circ \sqcup(\ t'\ ::\ {}^{<}t' \sqcap {}^{\geqslant}t' \circ \mathit{wipe}^{\circ}\ ) \\
= & \quad \{\ \text{cupjunctivity}\ \} \\
& \sqcup(\ t, t'\ ::\ ({}^{<}t \sqcap {}^{\geqslant}t \circ \mathit{wipe}^{\circ})\ \circ\ ({}^{<}t' \sqcap {}^{\geqslant}t' \circ \mathit{wipe}^{\circ})\ ) \\
= & \quad \{\ \text{Proposition 10.13a}\ \} \\
& \sqcup(\ t, t'\ ::\ ({}^{<}t \sqcup {}^{<}t') \ \sqcap\ ({}^{\geqslant}t \sqcap {}^{\geqslant}t') \circ \mathit{wipe}^{\circ}\ ) \\
= & \quad \{\ \text{Proposition 6.13c: diagonalisation}\ \} \\
& \sqcup(\ t''\ ::\ {}^{<}t'' \sqcap {}^{\geqslant}t'' \circ \mathit{wipe}^{\circ}\ ) \\
= & \quad \{\ \text{Characterisation 10.11}\ \} \\
& \mathit{cut}
\end{array}
$$

□

The interaction of *cut* with postcompose structures is even more difficult. There, the Axiom of Choice for Chronicles is needed. Observe the similarities between the antecedent and the consequent in the following result:

Proposition 10.14

a. $r^\circ \circ cut \sqsupseteq cut \circ r^\circ \Leftarrow r \bullet wipe \supseteq wipe \bullet r$

b. $r^\circ \circ cut \circ (r^\circ)_\succ \sqsubseteq cut \circ r^\circ \Leftarrow r \bullet wipe \bullet r_\succ \subseteq wipe \bullet r$

□

The second property is the more difficult one to prove. We need the Axiom of Choice 4.20. Because the proof is in terms of the model and not illuminating at all, it is relegated to Appendix B.
The condition $r \bullet wipe \supseteq wipe \bullet r$ is satisfied by, for example, strict relations. Informally, a strict relation r satisfies $a \langle r \rangle b \Rightarrow nm_a \langle r \rangle nm_b$, where nm_x is some tuple of nm with the same arity as x. That is, strictness of r implies $r \supseteq wipe \bullet r \bullet wipe^{-1}$. So for strict r we have that r° (weakly) commutes with *cut*:

$$
\begin{array}{ll}
 & strict.r \\
\Rightarrow & \quad \{ \text{ interpretation of a strict relation } \} \\
 & r \supseteq wipe \bullet r \bullet wipe^{-1} \\
= & \quad \{ \ wipe \text{ is a total function } \} \\
 & r \bullet wipe \supseteq wipe \bullet r \\
\Rightarrow & \quad \{ \text{ Proposition 10.14a } \} \\
 & r^\circ \circ cut \sqsupseteq cut \circ r^\circ
\end{array}
$$

10.3.2 Prefix defined

We go on with the proc *pref*, which uses the newly introduced proc *cut*:

Characterisation 10.15 *Prefix*

$$pref \triangleq pdly \circ (cut \sqcup I)$$

□

For $f \langle pref \rangle g$ we have that f is a (possibly) chopped, (possibly) delayed version of g. This proc *pref* will play a fundamental role in the definition of a buffer which delivers messages on request: when no more requests arrive, no more output will be generated, despite the fact that there might be more input. Then, the output is a prefix of the input.
The proc *pref* has a few nice properties:

Property 10.16 *Prefix*

a. $I \sqsubseteq pref$

b. $pdly \sqsubseteq pref$

c. $dly \sqsubseteq pref$

d. $pdly \circ pref = pref = pref \circ pdly$

e. $dly \circ pref = pref \circ dly$

f. $pref \circ pref = pref$

Proof:
The second property follows from the definition and from reflexivity of $cut \sqcup I$. Furthermore, 10.16a and 10.16c follow from 10.16b and Properties 10.5a and 10.10a. For the fourth we calculate:

$$
\begin{array}{ll}
 & pref \circ pdly \\
= & \quad \{ \text{ Characterisation 10.15 } \} \\
 & pdly \circ (cut \sqcup I) \circ pdly \\
= & \quad \{ \text{ cupjunctivity } \} \\
 & pdly \circ (cut \circ pdly \sqcup pdly) \\
= & \quad \{ \text{ Proposition 10.12c } \} \\
 & pdly \circ (pdly \circ cut \sqcup pdly) \\
= & \quad \{ \text{ cupjunctivity } \} \\
 & pdly \circ pdly \circ (cut \sqcup I) \\
= & \quad \{ \text{ Property 10.5b } \} \\
 & pdly \circ (cut \sqcup I) \\
= & \quad \{ \text{ Characterisation 10.15 } \} \\
 & pref
\end{array}
$$

Property 10.16e follows in a similar way. In fact, both sides equal $dly \circ (cut \sqcup I)$. Property 10.16f is derived from Proposition 10.13b
□

The more problematic property of *pref* is the commutation with postcompose. Fortunately, most of the work has been done in the previous section about *cut*:

Property 10.17 *Commutation with postcompose*

a. $r^\circ \circ pref \sqsupseteq pref \circ r^\circ \Leftarrow r \bullet wipe \supseteq wipe \bullet r$

b. $r^\circ \circ pref \circ (r^\circ)^\succ \sqsubseteq pref \circ r^\circ \Leftarrow r \bullet wipe \bullet r^\succ \subseteq wipe \bullet r$

c. $r^\circ \circ pref \circ (r^\circ)^\succ = pref \circ r^\circ \Leftarrow r \bullet wipe \bullet r^\succ = wipe \bullet r$

Proof:
Only statement 10.17a is shown:

$$
\begin{array}{ll}
 & r^\circ \circ pref \sqsupseteq pref \circ r^\circ \\
= & \quad \{ \text{ Characterisation 10.15 } \}
\end{array}
$$

$$
\begin{array}{ll}
 & r^\circ \circ pdly \circ (cut \sqcup I) \sqsupseteq pdly \circ (cut \sqcup I) \circ r^\circ \\
\Leftarrow & \quad \{ \text{ Property 10.5c; monotonicity } \} \\
 & r^\circ \circ (cut \sqcup I) \sqsupseteq (cut \sqcup I) \circ r^\circ \\
= & \quad \{ \text{ cupjunctivity } \} \\
 & r^\circ \circ cut \sqcup r^\circ \sqsupseteq cut \circ r^\circ \sqcup r^\circ \\
\Leftarrow & \quad \{ \text{ monotonicity } \} \\
 & r^\circ \circ cut \sqsupseteq cut \circ r^\circ \\
\Leftarrow & \quad \{ \text{ Proposition 10.14a } \} \\
 & r \bullet wipe \supseteq wipe \bullet r
\end{array}
$$

□

The total relation *es* commutes with *wipe*, Theorem 5.17e. Therefore, the total proc *eqsp*, Definition 5.16, commutes with *pref*. In the table below we record all combinations of the proc *eqsp* and the procs introduced in this chapter:

Theorem 10.18 *Combinations*

$P \circ Q$	Q	*eqsp*	*pdly*	*dly*	*pref*
P					
eqsp		*eqsp*	$pdly \circ eqsp$	$dly \circ eqsp$	$pref \circ eqsp$
pdly		$eqsp \circ pdly$	*pdly*	*dly*	*pref*
dly		$eqsp \circ dly$	*dly*	*dly*	$pref \circ dly$
pref		$eqsp \circ pref$	*pref*	$dly \circ pref$	*pref*

□

So much for the commutation properties of the four procs. Now we turn our attention to the interesting topic of causality.

10.4 Causality

In Chapters 7 and 9, the notions of causality and weak causality were introduced. It was motivated why these properties play an important role in distributed programming; in particular with respect to the feedback structure. Chapter 8 recorded causality properties of the constants of the calculus, and presented several preservation rules.
The three new primitives introduced in this chapter also satisfy the property of (weak) causality. The property that is proved is:

Property 10.19 *Causality of pdly, dly and pref*

a. *wcausal.pdly*

b. *causal.dly*

c. *wcausal.pref*

Proof:
To establish 10.19a, we first show that $^\circ\sigma$ (where $pd.\sigma$, Characterisation 10.1) is weakly causal in the sense of Characterisation 9.2. For $winert.^\circ\sigma$ we calculate for all t:

$$
\begin{array}{ll}
 & eqar \sqcap {}^\circ\sigma \circ {}^<t \circ ({}^\circ\sigma)^\cup \\
\sqsubseteq & \quad \{ \text{ monotonicity; Proposition 6.3f } \} \\
 & {}^\circ\sigma \circ {}^<t \circ {}^\circ(\sigma^{-1}) \\
= & \quad \{ \ \sigma^{-1} \text{ is an order-iso: Proposition 10.3a } \} \\
 & {}^\circ\sigma \circ {}^\circ(\sigma^{-1}) \circ {}^<(\sigma^{-1} \bullet t) \\
= & \quad \{ \text{ Proposition 6.3f } \} \\
 & {}^\circ\sigma \circ ({}^\circ\sigma)^\cup \circ {}^<(\sigma^{-1} \bullet t) \\
\sqsubseteq & \quad \{ \ \sigma \text{ is total: Proposition 6.4a, i.e., } {}^\circ\sigma \circ ({}^\circ\sigma)^\cup \sqsubseteq I \ \} \\
 & {}^<(\sigma^{-1} \bullet t) \\
\sqsubseteq & \quad \{ \ \sigma \subseteq \leqslant \ \equiv \ \sigma^{-1} \subseteq \geqslant \text{: Proposition 7.5 } \} \\
 & {}^<t
\end{array}
$$

For the property of closedness we remark that precompose structures never equal $\perp\!\!\!\perp$. The second conjunct of Characterisation 7.17 is verified next. It suffices to show totality of $^\circ\sigma$:

$$
\begin{array}{ll}
 & \top\!\!\top \circ {}^\circ\sigma \\
= & \quad \{ \text{ Theorem 3.6 } \} \\
 & \top\!\!\top \circ \top\!\!\top \circ {}^\circ\sigma \\
= & \quad \{ \ \top\!\!\top = \top^\circ \text{; Proposition 10.3c } \} \\
 & \top\!\!\top \circ {}^\circ\sigma \circ \top\!\!\top \\
= & \quad \{ \ {}^\circ\sigma \neq \perp\!\!\!\perp \text{: Cone Rule 3.19 } \} \\
 & \top\!\!\top
\end{array}
$$

The equality $\top\!\!\top \circ {}^\circ\sigma = \top\!\!\top$ equivales $({}^\circ\sigma)^> = I$. Finally, causality of $pdly$ in the sense of Characterisation 7.33 follows straightforwardly from Characterisation 10.4 and the causality of $^\circ\sigma$ derived above.
We already remarked that $dl.\tau \equiv arch.\tau^{-1}$. Therefore, to prove 10.19b, we show that τ^{-1} is a candidate for the archimedean function required in Characterisation 7.13:

$$
\begin{array}{ll}
 & eqar \sqcap {}^\circ\tau \circ {}^<t \circ ({}^\circ\tau)^\cup \\
\sqsubseteq & \quad \{ \text{ monotonicity; Proposition 6.3f } \} \\
 & {}^\circ\tau \circ {}^<t \circ {}^\circ(\tau^{-1}) \\
= & \quad \{ \ \tau^{-1} \text{ is an order-iso: Proposition 10.3a } \} \\
 & {}^\circ\tau \circ {}^\circ(\tau^{-1}) \circ {}^<(\tau^{-1} \bullet t) \\
\sqsubseteq & \quad \{ \text{ Proposition 6.3f; } \tau \text{ is total: Proposition 6.4a } \} \\
 & {}^<(\tau^{-1} \bullet t)
\end{array}
$$

This shows $inert.^\circ\tau$. The proof of closedness of $^\circ\tau$ and of causality of dly are copies of the corresponding proofs for $pdly$. The weak causality of $pref$ boils down to weak causality of cut:

$$
\begin{array}{ll}
 & wcausal.pref \\
= & \quad \{ \text{ Characterisation 10.15 } \}
\end{array}
$$

$$
\begin{array}{ll}
 & wcausal.(pdly \circ (cut \sqcup I)) \\
\Leftarrow & \quad \{\ pdly^{\rangle} = I,\ pdly \sqsubseteq eqar\text{: Axiom 9.8q}\ \} \\
 & wcausal.pdly \wedge wcausal.(cut \sqcup I) \\
\Leftarrow & \quad \{\ \text{Property 10.19a; } I^{\rangle} = I\text{: Axiom 9.8o}\ \} \\
 & wcausal.cut \wedge cut^{\rangle} = I \wedge wcausal.I \\
= & \quad \{\ \text{Theorem 7.7b}\ \} \\
 & wcausal.cut \wedge cut^{\rangle} = I
\end{array}
$$

By definition, cut can be decomposed into $({}^{<}t \sqcap {}^{\geqslant}t \circ wipe^{\circ})$-structures for all t. So it suffices to show, for all t, totality and causality of ${}^{<}t \sqcap {}^{\geqslant}t \circ wipe^{\circ}$ in the sense of Characterisation 7.20. Several properties of $wipe$ occur in the following calculations:

$$
\begin{array}{ll}
 & ({}^{<}t \sqcap {}^{\geqslant}t \circ wipe^{\circ})^{\rangle} \\
= & \quad \{\ (P \sqcap Q \circ R)^{\rangle} \ =\ (Q^{\cup} \circ P \sqcap R)^{\rangle}\ \} \\
 & ({}^{\geqslant}t)^{\cup} \circ {}^{<}t \sqcap wipe^{\circ})^{\rangle} \\
= & \quad \{\ \text{Propositions 6.11c, 6.11f and 6.11g}\ \} \\
 & (eqar \sqcap wipe^{\circ})^{\rangle} \\
= & \quad \{\ \text{corollary of Theorem 5.14: } wipe^{\circ} \sqsubseteq eqar\ \} \\
 & (wipe^{\circ})^{\rangle} \\
= & \quad \{\ \text{properties } wipe\text{: } wipe^{\circ} \text{ is total}\ \} \\
 & I
\end{array}
$$

Therefore, cut is total on I. Because I is closed, $I^{\rangle} = I$, it follows that cut is also closed. Finally, weak inertia of ${}^{<}t \sqcap {}^{\geqslant}t \circ wipe^{\circ}$ is proved for all t':

$$
\begin{array}{ll}
 & eqar \ \sqcap\ ({}^{<}t \sqcap {}^{\geqslant}t \circ wipe^{\circ}) \ \circ\ {}^{<}t' \ \circ\ ({}^{<}t \sqcap {}^{\geqslant}t \circ wipe^{\circ})^{\cup} \\
\sqsubseteq & \quad \{\ \text{monotonicity}\ \} \\
 & {}^{<}t \circ {}^{<}t' \circ {}^{<}t \sqcap {}^{\geqslant}t \circ wipe^{\circ} \circ {}^{<}t' \circ ({}^{\geqslant}t \circ wipe^{\circ})^{\cup} \\
\sqsubseteq & \quad \{\ \text{property } wipe\text{: } wipe^{\circ} \circ {}^{<}t' \sqsubseteq {}^{<}t' \circ wipe^{\circ}\text{; reverse}\ \} \\
 & {}^{<}t \circ {}^{<}t' \circ {}^{<}t \sqcap {}^{\geqslant}t \circ {}^{<}t' \circ wipe^{\circ} \circ (wipe^{\circ})^{\cup} \circ {}^{\geqslant}t \\
\sqsubseteq & \quad \{\ \text{property } wipe\text{: } wipe^{\circ} \text{ is a function}\ \} \\
 & {}^{<}t \circ {}^{<}t' \circ {}^{<}t \sqcap {}^{\geqslant}t \circ {}^{<}t' \circ {}^{\geqslant}t \\
= & \quad \{\ \text{Proposition 6.13a; Propositions 6.11f and 6.11b}\ \} \\
 & {}^{<}t \circ {}^{<}t' \sqcap {}^{\geqslant}t \circ {}^{<}t' \\
= & \quad \{\ \text{Characterisation 6.10; Proposition 6.8}\ \} \\
 & {}^{\circ}((<\bullet t)^{\prec} \bullet (<\bullet t')^{\prec}) \sqcap {}^{\circ}((\geqslant\bullet t)^{\prec} \bullet (<\bullet t')^{\prec}) \\
= & \quad \{\ \text{Proposition 6.3b; cupjunctivity}\ \} \\
 & {}^{\circ}(((<\bullet t)^{\prec} \cup (\geqslant\bullet t)^{\prec}) \bullet (<\bullet t')^{\prec}) \\
= & \quad \{\ \text{Proposition 6.9b}\ \} \\
 & {}^{\circ}((<\bullet t')^{\prec}) \\
= & \quad \{\ \text{Characterisation 6.10}\ \} \\
 & {}^{<}t'
\end{array}
$$

The property $wipe^{\circ} \circ {}^{<}t' \sqsubseteq {}^{<}t' \circ wipe^{\circ}$ is easily verified in the model
□

This concludes the introduction of the three primitive procs *pdly*, *dly* and *pref*. To relate the procs *pdly*, *pref* and *eqsp* more tightly to each other, two new properties are added to the calculus.

10.5 New properties

This section introduces two extra properties, connecting the three primitive procs *eqsp*, *pdly* and *pref*. The first property is needed for a correct interpretation of the synchronise proc, which will be defined in the next chapter:

Property 10.20 *Induction*

$$I \;=\; eqsp \sqcap pdly^{\cup} \circ pref$$

□

The property is implied by the assumption in the model that the support of a chronicle is well-ordered under $<$. Being well-ordered is in one-to-one correspondence with admitting induction. The details of the proof are omitted, but the key theorem is: if two well-ordered sets are isomorphic (implied by *eqsp*), then there exists only one order-isomorphism which establishes their similarity, see Cantor [Can97], who listed quite a few results about well-ordered sets, or Kuratowski & Mostowski [KM68]. In the above case, the mapping turns out to be a (partial) identity.
Property 10.20 can be rewritten, thereby removing the reverse. This rewriting gives more insight in the property:

$$\begin{array}{ll} & I \;=\; eqsp \sqcap pdly^{\cup} \circ pref \\ = & \quad \{\; eqsp, pdly \text{ and } pref \text{ are all reflexive; Characterisation 10.4} \;\} \\ & I \;\sqsupseteq\; eqsp \sqcap \sqcup(\, \sigma \;::\; {}^{\circ}\sigma\,)^{\cup} \circ pref \\ = & \quad \{\; \text{cupjunctivity properties; plat calculus} \;\} \\ & \forall(\sigma \;::\; I \;\sqsupseteq\; eqsp \sqcap ({}^{\circ}\sigma)^{\cup} \circ pref\,) \\ = & \quad \{\; {}^{\circ}\sigma \text{ and } ({}^{\circ}\sigma)^{\cup} \text{ are functions: monotonicity and Theorem 3.35b} \;\} \\ & \forall(\sigma \;::\; {}^{\circ}\sigma \;\sqsupseteq\; {}^{\circ}\sigma \circ eqsp \sqcap pref\,) \\ = & \quad \{\; eqsp \text{ is reflexive; } {}^{\circ}\sigma \sqsubseteq pref \;\} \\ & \forall(\sigma \;::\; {}^{\circ}\sigma \;=\; {}^{\circ}\sigma \circ eqsp \sqcap pref\,) \end{array}$$

We pointed out that *eqsp* abstracts from the messages, whereas $pdly^{\cup} \circ pdly$ abstracts from the exact timing. If we combine these two procs, we expect to recover the original chronicle. And indeed, we have that Property 10.20 is equivalent to

$$I \;=\; eqsp \sqcap pdly^{\cup} \circ pdly \quad \wedge \quad pdly \;=\; pref \sqcap pdly \circ eqsp$$

which implies

$$I \;=\; eqsp \sqcap pdly^{\cup} \circ pdly$$

This states that if two chronicles have the same support, and if there is a one-to-one connection between the messages (without reordering), then the two chronicles are equal.

Another corollary of Property 10.20 is:

$$I = eqsp \sqcap pref$$

It is unclear at present whether the conjunction of these last two corollaries is equivalent to Property 10.20. If true, this will give a nice partition into two properties about the individual procs *pdly* and *pref*.
A second additional property expresses that any two elements of $\mathcal{C}$ are comparable with respect to the number of messages they are carrying. First, two preorders on chronicles are defined:

Definition 10.21 *Pre-order*

a. $\leqslant_{\#} \triangleq pdly^{\cup} \circ eqsp \circ pref$

b. $\geqslant_{\#} \triangleq (\leqslant_{\#})^{\cup}$

□

The interpretation of proc $\leqslant_{\#}$ is: $f \langle \leqslant_{\#} \rangle g \equiv$ '$\#f \leqslant \#g$'. We put the interpretation with function # between quotes because this function is not formally defined for chronicles. Informally, it stands for the ordinal number of the support of the chronicle under consideration.

Property 10.22 *Total pre-order*

- $\leqslant_{\#} \sqcup \geqslant_{\#} = \top\top$

□

The property states that the pre-order $\leqslant_{\#}$ on the set of chronicles $\mathcal{C}$ is total. Again, the justification of this property is a theorem in Kuratowski & Mostowski [KM68]. It states that for two well-ordered sets it is either the case that they are isomorphic, or one is isomorphic to an initial segment of the other.
The property is not valid for $\mathcal{T}$ isomorphic to $\mathbf{Z}$: the proc $pdly^{\cup} \circ pdly$ is not capable to relate each pair of chronicles with isomorphic supports. This is due to the restrictive definition of *pdly*: the lessening order-isomorphisms in $\mathbf{Z}$ are all of the form $-n$ for some $n \in \mathbf{N}$.
The three procs *pdly*, *dly* and *pref* will play a fundamental role in defining new procs and stating properties. The proc *cut* was used only as a subcomponent of the proc *pref* and will not occur as an individual proc in the chapters that follow. Only the derived properties of the three procs will be used; the definitions (in terms of precompose and chronicles) are not important anymore. From now on, as announced before, the procs are our first-class citizens.

Chapter 11

Synchronise

In this chapter, we define and investigate a synchronisation proc. First, we give an explanation of the proc in the model. Then, the definition is motivated, step by step.

11.1 Synchronise explained

The proc *Sync* takes two chronicles f and g as input, and synchronises the messages that occur there. To explain the behaviour of *Sync*, suppose that the proc relates the chronicles f, g, h and j in the following way:

$$h \blacktriangle j \ \langle Sync \rangle \ f \blacktriangle g$$

The output consists of the synchronised pairs, that is, $sp.h = sp.j$. *Sync* produces as many pairs as possible (without duplicating or loosing messages). So, the number of synchronised output pairs is the minimum of the number of messages on both input channels. The proc *Sync* is non-deterministic in the sense that the output can be delayed: once two input messages are coupled into one pair, *Sync* can wait for a while before this pair is sent to the output. Messages on one channel keep their original order.

11.2 Synchronise defined

From the explanation we conclude that chronicle h is a prefix of chronicle f:
h gets messages from f (without duplication, dropping or reordering messages). Secondly, h can be delayed, partly due to the necessity that f has to be synchronised with (a slow) g, partly due the possibility that *Sync* slows down the coupled pairs. Furthermore, it can be the case that f contains more messages than g, in which case h does not get all the messages from chronicle f.
A similar argument justifies the observation that j is a prefix of g. A first try for the definition of *Sync* is:

$$Sync = pref \parallel pref$$

But we can do better. Notice that with this definition it is always possible that two chronicles containing no-messages are produced. These two chronicles are perfectly synchronised, and they are also prefixes of the two input chronicles. To refine this erroneous first guess assume that f contains more messages than g. Then, it follows that h is a *real* prefix of f, but j is a *delayed version* of g. This excludes the production of no messages if there is input. By symmetry we get a second guess:

$$Sync \;=\; pref \parallel pdly \sqcup pdly \parallel pref$$

The most important aspect of *Sync* is not captured, yet: the output pairs are not necessarily synchronised. For that reason we introduce the interface *synp*:

Definition 11.1 *Identity on synchronised pairs*

$$synp \triangleq (I \vartriangle eqsp)^{<}$$

□

In the model *synp* reads: $h \blacktriangle j \,\langle synp \rangle\, h \blacktriangle j \;\equiv\; sp.h = sp.j$. In words this means: *synp* is the identity on synchronised pairs. So finally, we can define the proc *Sync*:

Definition 11.2 *Synchronise*

$$Sync \;\triangleq\; synp \circ (pref \parallel pdly \sqcup pdly \parallel pref)$$

□

For completeness, the interpretation of *Sync* in the model is given:

$$\begin{array}{ll} & h \blacktriangle j \;\langle Sync \rangle\; f \blacktriangle g \\ \equiv & \\ & sp.h = sp.j \;\wedge\; ((h \,\langle pref \rangle\, f \wedge j \,\langle pdly \rangle\, g) \;\vee\; (h \,\langle pdly \rangle\, f \wedge j \,\langle pref \rangle\, g)) \end{array}$$

The following section will investigate this proc in detail.

11.3 Theorems

In the calculations that follow we will frequently use the fact that the proc *pdly* commutes with postcompose, Property 10.5c. In particular, *pdly* commutes with *synp* and the projections $\ll$ and $\gg$ (because $synp = ((\mathcal{I} \blacktriangle es)^{\prec})^{\circ}$, $\ll = \ll^{\circ}$ and $\gg = \gg^{\circ}$). Having stated these properties, the rest of this chapter mainly contains derivable theorems.

Theorem 11.3 *Synchronise*

a. $Sync \;\in\; synp \sim I \parallel I$

b. $synp \sqsubseteq Sync$

c. $pdly \circ Sync = Sync \circ pdly = Sync$

d. $Sync \circ I \parallel pdly \;=\; Sync \circ pdly \parallel I \;=\; Sync \circ pdly \parallel pdly \;=\; Sync$

Proof:
The first theorem is almost too trivial:

$$
\begin{array}{ll}
 & synp \circ Sync \circ I \parallel I \\
= & \quad \{ \text{ Definition 11.2 } \} \\
 & synp \circ synp \circ (pref \parallel pdly \sqcup pdly \parallel pref) \circ I \parallel I \\
= & \quad \{ \text{ } synp \text{ is an interface: } synp \circ synp = synp \text{; cupjunctivity } \} \\
 & synp \circ (pref \parallel pdly \circ I \parallel I \sqcup pdly \parallel pref \circ I \parallel I) \\
= & \quad \{ \text{ Parallel-parallel fusion 3.12b; } I \text{ is the identity } \} \\
 & synp \circ (pref \parallel pdly \sqcup pdly \parallel pref) \\
= & \quad \{ \text{ Definition 11.2 } \} \\
 & Sync
\end{array}
$$

The second theorem, which states reflexivity of $Sync$ on $synp$, is proved as follows:

$$
\begin{array}{ll}
 & Sync \\
= & \quad \{ \text{ Definition 11.2 } \} \\
 & synp \circ (pref \parallel pdly \sqcup pdly \parallel pref) \\
\sqsupseteq & \quad \{ \text{ Properties 10.5a and 10.16a } \} \\
 & synp \circ (I \parallel I \sqcup I \parallel I) \\
= & \quad \{ \text{ } synp \sqsubseteq I \parallel I \text{: interfaces } \} \\
 & synp
\end{array}
$$

The proofs of the third and fourth statements exploit the corresponding absorption properties of $pref$ and $pdly$:

$$
\begin{array}{ll}
 & pdly \circ Sync \\
= & \quad \{ \text{ Definition 11.2 } \} \\
 & pdly \circ synp \circ (pref \parallel pdly \sqcup pdly \parallel pref) \\
= & \quad \{ \text{ } synp = ((\mathcal{I} \blacktriangle es)\prec)^{\circ} \text{: Property 10.5c; cupjunctivity } \} \\
 & synp \circ (pdly \circ pref \parallel pdly \sqcup pdly \circ pdly \parallel pref) \\
\sqsubseteq & \quad \{ \text{ Property 10.5d } \} \\
 & synp \circ (pdly \parallel pdly \circ pref \parallel pdly \sqcup pdly \parallel pdly \circ pdly \parallel pref) \\
= & \quad \{ \text{ Parallel-parallel fusion 3.12b } \} \\
 & synp \circ ((pdly \circ pref) \parallel (pdly \circ pdly) \sqcup (pdly \circ pdly) \parallel (pdly \circ pref)) \\
= & \quad \{ \text{ Properties 10.5b and 10.16d } \} \\
 & synp \circ (pref \parallel pdly \sqcup pdly \parallel pref) \\
= & \quad \{ \text{ Definition 11.2 } \} \\
 & Sync \\
\sqsubseteq & \quad \{ \text{ Property 10.5a } \} \\
 & pdly \circ Sync
\end{array}
$$

The proof of the second equality of statement 11.3c follows the same structure. For the fourth proof obligation we restrict our attention to the equality $Sync \circ pdly \parallel pdly = Sync$,

from which all the other equalities follow by reflexivity or idempotency of $pdly$:

$$
\begin{array}{ll}
 & Sync \circ pdly \parallel pdly \\
= & \quad \{ \text{ Definition 11.2 } \} \\
 & synp \circ (pref \parallel pdly \sqcup pdly \parallel pref) \circ pdly \parallel pdly \\
= & \quad \{ \text{ cupjunctivity; Parallel-parallel fusion 3.12b } \} \\
 & synp \circ ((pref \circ pdly) \parallel (pdly \circ pdly) \sqcup (pdly \circ pdly) \parallel (pref \circ pdly)) \\
= & \quad \{ \text{ Properties 10.5b and 10.16d } \} \\
 & synp \circ (pref \parallel pdly \sqcup pdly \parallel pref) \\
= & \quad \{ \text{ Definition 11.2 } \} \\
 & Sync
\end{array}
$$

□

To prove a number of other theorems, we use Property 10.20. The property is fundamental for a correct interpretation of the synchronise proc. Exploiting this property we can prove the idempotency of the proc $Sync$. As a preliminary result it is shown that the only action of $Sync$ on a chronicle carrying synchronised pairs is a possible delay:

Theorem 11.4 *Synchronise*

a. $Sync \circ synp = pdly \circ synp$

b. $Sync \circ Sync = Sync$

Proof:
Proving the first statement boils down to proving that $synp \circ pref \parallel pdly \circ synp$ is equal to $pdly \circ synp$. First, two subexpressions are evaluated:

$$
\begin{array}{ll}
 & pref \parallel pdly \\
= & \quad \{ \text{ Definition 3.10 } \} \\
 & \ll^{\cup} \circ pref \circ \ll \sqcap \gg^{\cup} \circ pdly \circ \gg \\
= & \quad \{ \ \gg^{\cup} = (\gg^{-1})^{\circ} \text{: Property 10.5c } \} \\
 & \ll^{\cup} \circ pref \circ \ll \sqcap pdly \circ \gg^{\cup} \circ \gg \\
\sqsubseteq & \quad \{ \text{ Dedekind } \} \\
 & pdly \circ (pdly^{\cup} \circ \ll^{\cup} \circ pref \circ \ll \sqcap \gg^{\cup} \circ \gg) \\
= & \quad \{ \ \ll = \ll^{\circ} \text{: Property 10.5c and reverse } \} \\
 & pdly \circ (\ll^{\cup} \circ pdly^{\cup} \circ pref \circ \ll \sqcap \gg^{\cup} \circ \gg) \\
= & \quad \{ \text{ Definition 3.10 } \} \\
 & pdly \circ (pdly^{\cup} \circ pref) \parallel I
\end{array}
$$

In the next calculation, we use the fact that the procs $synp \circ \ll^{\cup} \circ eqsp$ and $synp \circ \gg^{\cup} \circ eqsp$

are equal (because they are both equal to $eqsp \vartriangle eqsp$):

$$
\begin{array}{ll}
 & synp \circ eqsp \parallel I \circ synp \\
= & \quad \{ \text{ Definition 3.10 } \} \\
 & synp \circ (\ll^{\cup} \circ eqsp \circ \ll \sqcap \gg^{\cup} \circ \gg) \circ synp \\
= & \quad \{ \text{ } synp \text{ is an interface: Corollary 3.27c } \} \\
 & synp \circ \ll^{\cup} \circ eqsp \circ \ll \circ synp \sqcap synp \circ \gg^{\cup} \circ \gg \circ synp \\
= & \quad \{ \text{ } synp \circ \ll^{\cup} \circ eqsp = synp \circ \gg^{\cup} \circ eqsp \text{ } \} \\
 & synp \circ \gg^{\cup} \circ eqsp \circ \gg \circ synp \sqcap synp \circ \gg^{\cup} \circ \gg \circ synp \\
= & \quad \{ \text{ Theorem 5.18a: monotonicity } \} \\
 & synp \circ \gg^{\cup} \circ \gg \circ synp \\
= & \quad \{ \text{ Definition 3.10 } \} \\
 & synp \circ \top\top \parallel I \circ synp
\end{array}
$$

from which it follows that $synp \circ P \parallel I \circ synp = synp \circ (eqsp \sqcap P) \parallel I \circ synp$. Then:

$$
\begin{array}{ll}
 & synp \circ pref \parallel pdly \circ synp \\
\sqsubseteq & \quad \{ \text{ first calculation above } \} \\
 & synp \circ pdly \circ (pdly^{\cup} \circ pref) \parallel I \circ synp \\
= & \quad \{ \text{ } synp = ((\mathcal{I} \blacktriangle es)_{\prec})^{\circ} \text{: Property 10.5c } \} \\
 & pdly \circ synp \circ (pdly^{\cup} \circ pref) \parallel I \circ synp \\
= & \quad \{ \text{ second calculation above } \} \\
 & pdly \circ synp \circ (eqsp \sqcap pdly^{\cup} \circ pref) \parallel I \circ synp \\
= & \quad \{ \text{ Property 10.20 } \} \\
 & pdly \circ synp \circ I \parallel I \circ synp \\
= & \quad \{ \text{ } synp \sqsubseteq I \parallel I \text{: interfaces } \} \\
 & pdly \circ synp \\
= & \quad \{ \text{ } synp \text{ is an interface; Property 10.5c } \} \\
 & synp \circ pdly \circ synp \\
\sqsubseteq & \quad \{ \text{ Property 10.5d } \} \\
 & synp \circ pdly \parallel pdly \circ synp \\
\sqsubseteq & \quad \{ \text{ Property 10.16b } \} \\
 & synp \circ pref \parallel pdly \circ synp
\end{array}
$$

The equality $synp \circ pdly \parallel pref \circ synp = pdly \circ synp$ is derived in a symmetrical way. This proves statement 11.4a. And finally:

$$
\begin{array}{ll}
 & Sync \circ Sync \\
= & \quad \{ \text{ Theorem 11.3a } \} \\
 & Sync \circ synp \circ Sync \\
= & \quad \{ \text{ Theorem 11.4a } \} \\
 & pdly \circ synp \circ Sync \\
= & \quad \{ \text{ Theorem 11.3a } \} \\
 & pdly \circ Sync \\
= & \quad \{ \text{ Theorem 11.3c } \} \\
 & Sync
\end{array}
$$

□

The inclusion $synp \circ pref \parallel pdly \circ synp \sqsupseteq pdly \circ synp$ does not depend on Property 10.20. This suggests that one inclusion of Theorem 11.4a is independent of this property. And indeed, we did already prove $Sync \circ synp \sqsupseteq pdly \circ synp$, because this inclusion is equivalent to $Sync \sqsupseteq synp$, Theorem 11.3b.
The last calculations involve the typing of *Sync*. What is still missing is the totality of *Sync* on $I \parallel I$. It turns out that we need Property 10.22. We can derive, exploiting this property, the totality of proc *Sync* on pairs of chronicles:

Theorem 11.5 *Typing of Synchronise*

- $Sync \in I \rightsquigarrow I \parallel I$

Proof:
The proof involves several theorems about domains:

$$
\begin{array}{ll}
 & (synp \circ pref \parallel pdly)\rangle \\
= & \quad \{ \text{ Definition 11.1 } \} \\
 & ((I \vartriangle eqsp)\langle \circ pref \parallel pdly)\rangle \\
= & \quad \{ \text{ reverse: } P\rangle = (P^{\cup})\langle \text{, Theorems 3.11a and 3.26a } \} \\
 & (pref^{\cup} \parallel pdly^{\cup} \circ (I \vartriangle eqsp)\langle)\langle \\
= & \quad \{ \ (P \circ Q\langle)\langle = (P \circ Q)\langle \ \} \\
 & (pref^{\cup} \parallel pdly^{\cup} \circ I \vartriangle eqsp)\langle \\
= & \quad \{ \text{ Parallel-split fusion 3.12a } \} \\
 & (pref^{\cup} \vartriangle (pdly^{\cup} \circ eqsp))\langle \\
= & \quad \{ \ (P \vartriangle Q)\langle \ = \ (I \vartriangle (Q \circ P^{\cup}))\langle \ \} \\
 & (I \vartriangle (pdly^{\cup} \circ eqsp \circ pref))\langle \\
= & \quad \{ \text{ Definition 10.21 } \} \\
 & (I \vartriangle \leqslant_{\#})\langle
\end{array}
$$

The equality $(synp \circ pdly \parallel pref)\rangle = (I \vartriangle \geqslant_{\#})\langle$ is derived in the same way.
Now the proof of $Sync\rangle = I \parallel I$ can be shown:

$$
\begin{array}{ll}
 & Sync\rangle \\
= & \quad \{ \text{ Definition 11.2 } \} \\
 & (synp \circ (pref \parallel pdly \sqcup pdly \parallel pref))\rangle \\
= & \quad \{ \text{ cupjunctivity of composition and domain } \} \\
 & (synp \circ pref \parallel pdly)\rangle \sqcup (synp \circ pdly \parallel pref)\rangle \\
= & \quad \{ \text{ calculation above } \} \\
 & (I \vartriangle \leqslant_{\#})\langle \sqcup (I \vartriangle \geqslant_{\#})\langle \\
= & \quad \{ \text{ cupjunctivity domain and } I \vartriangle \ \} \\
 & (I \vartriangle (\leqslant_{\#} \sqcup \geqslant_{\#}))\langle \\
= & \quad \{ \text{ Property 10.22 } \} \\
 & (I \vartriangle \top\top)\langle \\
= & \quad \{ \text{ Theorems 3.11b and 3.40e } \} \\
 & I \parallel I
\end{array}
$$

□

In fact, Theorem 11.5 equivales Property 10.22, which follows from the equivalence:

$$(I \triangle P)^{<} = (I \triangle Q)^{<} \;\equiv\; P = Q$$

This justifies the remark that 'we need Property 10.22'.
A result of Theorems 11.3a, 11.3b and 11.5 is the typing of proc *Sync*:

Corollary 11.6 *Typing Synchronise*

- $$Sync \;\in\; synp \rightsquigarrow I \parallel I$$

□

To conclude, we handle the commutation of *Sync* with postcompose structures. Because *Sync* is built from the procs *pdly*, *pref* and *synp* , the commutation of these procs with postcompose structures will be needed. For *pdly* and *pref* we have those results: Property 10.5c and Property 10.17. The corresponding property for *synp* is still missing. For later use we prove a somewhat stronger result than the one needed in this section:

Theorem 11.7 *Equivalences*
The following three statements are all (pairwise) equivalent:

a. $r^{\circ} \sqsubseteq eqsp$

b. $synp \circ r^{\circ} \parallel I \;=\; r^{\circ} \parallel I \circ synp$

c. $eqsp \circ r^{\circ} = eqsp \circ (r^{\circ})^{>}$

Proof:
The proof is by mutual implication. To prove the implication 11.7a ⇒ 11.7b a lemma is given first. It states that the specific postcompose structure $eqsp \parallel I$ commutes with the interface *synp* :

$$\begin{array}{ll} & synp \circ eqsp \parallel I \\ = & \quad \{ \text{ Definition 3.10 } \} \\ & synp \circ (\ll^{\cup} \circ eqsp \circ \ll \sqcap \gg^{\cup} \circ \gg) \\ = & \quad \{ \text{ interfaces; } synp \circ \ll^{\cup} \circ eqsp = synp \circ \gg^{\cup} \circ eqsp \text{ } \} \\ & synp \circ (\gg^{\cup} \circ eqsp \circ \ll \sqcap \gg^{\cup} \circ \gg) \\ = & \quad \{ \gg^{\cup} \text{ is an injection: Theorem 3.35a } \} \\ & synp \circ \gg^{\cup} \circ (eqsp \circ \ll \sqcap \gg) \\ = & \quad \{ synp \circ \gg^{\cup} = eqsp \triangle I \text{; Definition 3.10, reverse } \} \\ & eqsp \triangle I \circ (eqsp \triangle I)^{\cup} \\ = & \quad \{ \text{ same steps backwards, reversed } \} \\ & eqsp \parallel I \circ synp \end{array}$$

Then, this is generalised to arbitrary postcompose structures below *eqsp*, thus establishing 11.7a ⇒ 11.7b:

$$\begin{array}{ll} & synp \circ r^{\circ} \parallel I \\ = & \quad \{ \text{ assumption } \} \end{array}$$

$$
\begin{array}{ll}
 & synp \circ (r^\circ \parallel I \sqcap eqsp \parallel I) \\
= & \quad \{ \ synp \text{ is an interface: Theorem 3.26b} \ \} \\
 & r^\circ \parallel I \sqcap synp \circ eqsp \parallel I \\
= & \quad \{ \ \text{lemma above} \ \} \\
 & r^\circ \parallel I \sqcap eqsp \parallel I \circ synp \\
= & \quad \{ \ synp \text{ is an interface: Theorem 3.26b} \ \} \\
 & (r^\circ \parallel I \sqcap eqsp \parallel I) \circ synp \\
= & \quad \{ \ \text{assumption} \ \} \\
 & r^\circ \parallel I \circ synp
\end{array}
$$

For the implication 11.7b $\Rightarrow$ 11.7c we calculate:

$$
\begin{array}{ll}
 & synp \circ r^\circ \parallel I \ = \ r^\circ \parallel I \circ synp \\
\Rightarrow & \quad \{ \ \text{Leibniz} \ \} \\
 & \gg \circ synp \circ r^\circ \parallel I \circ I \vartriangle \top\top \ = \ \gg \circ r^\circ \parallel I \circ synp \circ I \vartriangle \top\top \\
= & \quad \{ \ \gg \circ synp = (eqsp \vartriangle I)^\cup ; \ synp \circ I \vartriangle \top\top \ = \ I \vartriangle eqsp \ \} \\
 & (eqsp \vartriangle I)^\cup \circ r^\circ \parallel I \circ I \vartriangle \top\top \ = \ \gg \circ r^\circ \parallel I \circ I \vartriangle eqsp \\
= & \quad \{ \ \text{Parallel-split fusion 3.12a} \ \} \\
 & (eqsp \vartriangle I)^\cup \circ r^\circ \vartriangle \top\top \ = \ \gg \circ r^\circ \vartriangle eqsp \\
= & \quad \{ \ \text{Axiom 3.10d; Split computation 3.32b} \ \} \\
 & eqsp \circ r^\circ \ = \ eqsp \circ (r^\circ)^\rhd
\end{array}
$$

And the implication 11.7c $\Rightarrow$ 11.7a follows straightforwardly:

$$
\begin{array}{ll}
 & eqsp \circ r^\circ \ = \ eqsp \circ (r^\circ)^\rhd \\
\Rightarrow & \quad \{ \ \text{Theorem 3.30d} \ \} \\
 & eqsp \circ r^\circ \sqsubseteq eqsp \\
\Rightarrow & \quad \{ \ \text{Theorem 5.18a} \ \} \\
 & r^\circ \sqsubseteq eqsp
\end{array}
$$

□

The following calculation gives more insight in the condition $r^\circ \sqsubseteq eqsp$. It relates the condition to a transformation property of r which is easier to check:

Theorem 11.8 *Postcompose*

$$r^\circ \sqsubseteq eqsp \ \Leftarrow \ sm \bullet r = r \bullet sm$$

Proof:

$$
\begin{array}{ll}
 & r^\circ \sqsubseteq eqsp \\
= & \quad \{ \ \text{Definition 5.16} \ \} \\
 & r^\circ \ \sqsubseteq \ (sm \setminus (\top \bullet sm) \cap (sm \bullet \top) / sm)^\circ \\
\Leftarrow & \quad \{ \ \text{monotonicity of postcompose} \ \} \\
 & r \ \subseteq \ sm \setminus (\top \bullet sm) \cap (sm \bullet \top) / sm \\
= & \quad \{ \ \text{plat calculus} \ \} \\
 & r \subseteq sm \setminus (\top \bullet sm) \ \wedge \ r \subseteq (sm \bullet \top) / sm \\
= & \quad \{ \ \text{Definition 3.4} \ \}
\end{array}
$$

$$
\begin{array}{ll}
 & sm \bullet r \subseteq \top \bullet sm \ \wedge \ r \bullet sm \subseteq sm \bullet \top \\
\Leftarrow & \qquad \{ \ r \subseteq \top \ \} \\
 & sm \bullet r = r \bullet sm
\end{array}
$$

□

The last implication in the proof is actually an equivalence. This shows that es, Definition 5.16a, is the greatest relation solving the equation $r \ :: \ sm \bullet r = r \bullet sm$.
The next lemma, which will be used in the final chapter, shows an application:

Lemma 11.9

$$eqsp \circ I \vartriangle eqsp \ = \ eqsp$$

Proof:

$$
\begin{array}{ll}
 & eqsp \circ I \vartriangle eqsp \ = \ eqsp \\
= & \qquad \{ \ I \vartriangle eqsp = (\mathcal{I} \blacktriangle es)^\circ ; (I \vartriangle eqsp)_> = I \ \} \\
 & eqsp \circ (\mathcal{I} \blacktriangle es)^\circ \ = \ eqsp \circ ((\mathcal{I} \blacktriangle es)^\circ)_> \\
\Leftarrow & \qquad \{ \ \text{Theorems 11.7 and 11.8} \ \} \\
 & sm \bullet \mathcal{I} \blacktriangle es \ = \ \mathcal{I} \blacktriangle es \bullet sm \\
= & \qquad \{ \ \text{Definition 4.4: cupjunctivity; Definition 4.3: } msg \bullet \mathcal{I} \times \mathcal{I} \ = \ \bot \ \} \\
 & sm \times \mathcal{I} \bullet \mathcal{I} \blacktriangle es \ \cup \ \mathcal{I} \times sm \bullet \mathcal{I} \blacktriangle es \ = \ \mathcal{I} \blacktriangle es \bullet sm \ \cup \ \mathcal{I} \blacktriangle es \bullet sm \\
= & \qquad \{ \ \text{Product-split fusion 3.12a; } sm \text{ is identity: distribution} \ \} \\
 & sm \blacktriangle es \ \cup \ \mathcal{I} \blacktriangle (sm \bullet es) \ = \ sm \blacktriangle es \ \cup \ \mathcal{I} \blacktriangle (es \bullet sm) \\
= & \qquad \{ \ \text{Theorem 5.17d} \ \} \\
 & True
\end{array}
$$

□

Now we can state and prove the commutation property of *Sync* with postcompose:

Theorem 11.10 *Commutation with postcompose*

a.
$$
\begin{array}{ll}
 & r^\circ \parallel s^\circ \circ Sync \ \sqsupseteq \ Sync \circ r^\circ \parallel s^\circ \\
\Leftarrow & \\
 & r \bullet sm = sm \bullet r \ \wedge \ r \bullet wipe \supseteq wipe \bullet r \\
 & \ \wedge \ s \bullet sm = sm \bullet s \ \wedge \ s \bullet wipe \supseteq wipe \bullet s
\end{array}
$$

b.
$$
\begin{array}{ll}
 & r^\circ \parallel s^\circ \circ Sync \circ (r^\circ \parallel s^\circ)_> \ \sqsubseteq \ Sync \circ r^\circ \parallel s^\circ \\
\Leftarrow & \\
 & r \bullet sm = sm \bullet r \ \wedge \ r \bullet wipe \bullet r_\succ \subseteq wipe \bullet r \\
 & \ \wedge \ s \bullet sm = sm \bullet s \ \wedge \ s \bullet wipe \bullet s_\succ \subseteq wipe \bullet s
\end{array}
$$

c.
$$
\begin{array}{ll}
 & r^\circ \parallel s^\circ \circ Sync \circ (r^\circ \parallel s^\circ)_> \ = \ Sync \circ r^\circ \parallel s^\circ \\
\Leftarrow & \\
 & r \bullet sm = sm \bullet r \ \wedge \ r \bullet wipe \bullet r_\succ = wipe \bullet r \\
 & \ \wedge \ s \bullet sm = sm \bullet s \ \wedge \ s \bullet wipe \bullet s_\succ = wipe \bullet s
\end{array}
$$

Proof:
We only prove the first theorem:

$$
\begin{array}{ll}
 & r^\circ \parallel s^\circ \circ \mathit{Sync} \\
= & \quad \{ \text{ Definition 11.2 } \} \\
 & r^\circ \parallel s^\circ \circ \mathit{synp} \circ (\mathit{pref} \parallel \mathit{pdly} \sqcup \mathit{pdly} \parallel \mathit{pref}) \\
= & \quad \{ \text{ assumptions } r \bullet \mathit{sm} = \mathit{sm} \bullet r \text{: Theorems 11.8 and 11.7 } \} \\
 & \mathit{synp} \circ r^\circ \parallel s^\circ \circ (\mathit{pref} \parallel \mathit{pdly} \sqcup \mathit{pdly} \parallel \mathit{pref}) \\
= & \quad \{ \text{ cupjunctivity; Parallel-parallel fusion 3.12b } \} \\
 & \mathit{synp} \circ ((r^\circ \circ \mathit{pref}) \parallel (s^\circ \circ \mathit{pdly}) \sqcup (r^\circ \circ \mathit{pdly}) \parallel (s^\circ \circ \mathit{pref})) \\
= & \quad \{ \text{ Property 10.5c } \} \\
 & \mathit{synp} \circ ((r^\circ \circ \mathit{pref}) \parallel (\mathit{pdly} \circ s^\circ) \sqcup (\mathit{pdly} \circ r^\circ) \parallel (s^\circ \circ \mathit{pref})) \\
\sqsupseteq & \quad \{ \text{ assumptions } r \bullet \mathit{wipe} \supseteq \mathit{wipe} \bullet r \text{: Property 10.17a } \} \\
 & \mathit{synp} \circ ((\mathit{pref} \circ r^\circ) \parallel (\mathit{pdly} \circ s^\circ) \sqcup (\mathit{pdly} \circ r^\circ) \parallel (\mathit{pref} \circ s^\circ)) \\
= & \quad \{ \text{ Parallel-parallel fusion 3.12b; cupjunctivity } \} \\
 & \mathit{synp} \circ (\mathit{pref} \parallel \mathit{pdly} \sqcup \mathit{pdly} \parallel \mathit{pref}) \circ r^\circ \parallel s^\circ \\
= & \quad \{ \text{ Definition 11.2 } \} \\
 & \mathit{Sync} \circ r^\circ \parallel s^\circ
\end{array}
$$

□

We conclude the examination of commutation properties of *Sync* with the remark that $r \bullet \mathit{sm} = \mathit{sm} \bullet r$ stands in no logical relation with either the inclusion $r \bullet \mathit{wipe} \supseteq \mathit{wipe} \bullet r$ or $r \bullet \mathit{wipe} \bullet r_\succ \subseteq \mathit{wipe} \bullet r$. The last difficult task is to show the weak causality of *Sync*:

Property 11.11 *Weak causality of Sync*

- $\mathit{Sync} \in \mathit{synp}' \overset{w}{\rightsquigarrow} (I \parallel I)'$

Proof:
The totality and surjectivity of *Sync* are recorded in Corollary 11.6. Primed typing follows from the corresponding properties of *pdly* and *pref*:

$$
\begin{array}{ll}
 & \mathit{Sync} \in I' \sim I' \\
= & \quad \{ \text{ Definition 11.2 } \} \\
 & \mathit{synp} \circ (\mathit{pref} \parallel \mathit{pdly} \sqcup \mathit{pdly} \parallel \mathit{pref}) \ \in \ I' \sim I' \\
\Leftarrow & \quad \{ \ A \circ P \in I' \sim I' \ \Leftarrow \ P \in I' \sim I' \ \} \\
 & \mathit{pref} \parallel \mathit{pdly} \sqcup \mathit{pdly} \parallel \mathit{pref} \ \in \ I' \sim I' \\
\Leftarrow & \quad \{ \text{ preservation by parallel composition and cup } \} \\
 & \mathit{pref} \in I' \sim I' \ \wedge \ \mathit{pdly} \in I' \sim I' \\
= & \quad \{ \ \mathit{pdly} \sqsubseteq \mathit{pref} \sqsubseteq \mathit{eqar} \text{: Lemma 7.10a } \} \\
 & \mathit{True}
\end{array}
$$

It follows from Axioms 9.8o and 9.8r and Property 10.19 that $\mathit{pref} \parallel \mathit{pdly} \sqcup \mathit{pdly} \parallel \mathit{pref}$ is weakly causal. Furthermore, interfaces are weakly causal whenever they are closed. In particular, non-empty interfaces of the form a° are closed. Because the non-empty interface *synp* can be written as $((\mathcal{I} \blacktriangleleft \mathit{es})_\prec)^\circ$ it is also closed, and therefore weakly causal.

However, Axiom 9.8q is not applicable because *synp* is too restrictive: $synp \sqsupseteq I \parallel I$ is a false statement.
Consider the refinement *sync* of *Sync*: it is the proc which is as productive as possible, i.e., produces a pair as soon as it is able to. The proc *Sync* can be obtained by composing *sync* with *pdly*: $Sync = pdly \circ sync$. The proc *sync* is weakly causal in the sense of Characterisation 9.2a because it reacts instantaneously, and therefore it is weakly causal according to Characterisation 9.2b. It follows from Axiom 9.8q that the proc *Sync* is weakly causal. □

We now turn to the proc *Buff*. This proc is the main instrument to build buffered procs, which permits asynchronous communication in a calculus that is basically synchronous.

Chapter 12

Buffer

We characterise distributed programs by their input-output behaviour. On the level of the model, there is no buffering. A result of this design decision (on the level of designing a calculus) is that the process of buffering has to be made explicit by a proc *Buff*.

12.1 Buffer defined

Our buffer has two input channels and one output channel. On the first input channel the messages come in; on the second input channel requests for new messages arrive. The principle is that each request signal is coupled with a message. This action is performed by the synchronisation proc *Sync* which was introduced in the previous chapter. After that, the request signals are stripped of by a projection, and the corresponding messages are delivered on the output channel. There are no type restrictions on the request signals: only the *occurrence* of a request is important, not the contents. This motivates the following definition:

Definition 12.1 *Buffer*

$$Buff \triangleq \ll \circ Sync$$

□

The proc is defined in terms of other procs. In Lemma 12.3 we will derive an expression in terms of the primitives of the calculus. But still, no reference to chronicles or messages occurs in the definition: procs are first-class citizens. Several other approaches define buffers or queues for every possible input with the explicit mentioning of (streams of) data elements. For example, in Hoare [Hoa85]:

$$BUFFER = P_{\langle\rangle}$$

where

$$P_{\langle\rangle} = left?x \to P_{\langle x\rangle}$$

and

$$P_{\langle x\rangle \frown s} = (left?y \to P_{\langle x\rangle \frown s \frown \langle y\rangle} \mid right!x \to P_s)$$

The input channel is *left*, the output channel is *right*, and x and y are data elements. Notice the explicit state of the buffer, modelled as a joinlist s. This buffer behaves like a queue: messages join the right-hand end of the queue and leave it from the left end, in the same order as they joined, but after a possible delay, during which later messages may join the queue. In that perspective, this buffer is similar to our proc *pdly*. In Baeten & Weijland [BW90] a similar definition is found:

$$Q = Q_\lambda = \sum_{d \in D} r_1(d) \cdot Q_d$$

and

$$Q_{\sigma d} = s_2(d) \cdot Q_\sigma + \sum_{e \in D} r_1(e) \cdot Q_{e \sigma d} \quad \text{(for every } \sigma \in D^* \text{ and } d \in D\text{)}$$

There are no request signals in the two approaches. In Broy [Bro90] a specification $QS.f$ for queue f with request signals ¿ is defined:

$$f = h.\langle\rangle$$

where

$$(h.\langle\rangle).(¿\hat{}x) = ¿\hat{}(h.\langle\rangle).x$$

,

$$(h(d\hat{}q)).(¿\hat{}x) = d\hat{}(h.q).x$$

and

$$(h.q).(d\hat{}x) = (h(q\hat{}d)).x$$

The identifier x is a finite or infinite list of data elements and request signals, d is an ordinary data element, q is a finite list of data elements, and $\hat{}$ is the append operator. The messages leave the queue on request in the same order as they entered. In this sense the queue behaves like our buffer. However, a request sent to an empty queue results in the output of that request signal as an error message; this contrasts with our approach: *Buff* simply goes to a state with an extra pending request.

12.2 Theorems

Next, a number of theorems for the buffer *Buff* are recorded. The most significant results are those involving the feedback operator ϖ, Theorem 12.7. The theorems of *Sync* give rise to a set of theorems of *Buff*:

Theorem 12.2 *Buffer*

- a. $Buff \in I \rightsquigarrow I \parallel I$

b. $pdly \circ Buff = Buff \circ pdly = Buff$

c. $Buff \circ I \parallel pdly = Buff \circ pdly \parallel I = Buff \circ pdly \parallel pdly = Buff$

d. $Buff \circ synp = pdly \circ \ll \circ synp$

□

A lemma which turns out to be very useful expresses the buffer *Buff* in terms of the primitives. We could have taken this as its definition, but the partition into *Sync* and a projection looks nicer:

Lemma 12.3

$$Buff \;=\; (pref \circ \ll \sqcap\; eqsp \circ pdly \circ \gg) \;\sqcup\; (pdly \circ \ll \sqcap\; eqsp \circ pref \circ \gg)$$

Proof:
In the proof that follows we will use the fact $\ll \circ\, synp = (I \vartriangle eqsp)^{\cup}$. This is a consequence of the equality $(I \vartriangle P)^{<} \circ I \vartriangle \top\top \;=\; I \vartriangle P$.

$$\begin{array}{ll}
 & Buff \\
= & \qquad \{ \text{ Definitions 12.1 and 11.2 } \} \\
 & \ll \circ\; synp \;\circ\; (pref \parallel pdly \sqcup pdly \parallel pref) \\
= & \qquad \{ \;\ll \circ\, synp = (I \vartriangle eqsp)^{\cup} \; \} \\
 & (I \vartriangle eqsp)^{\cup} \;\circ\; (pref \parallel pdly \sqcup pdly \parallel pref) \\
= & \qquad \{ \text{ cupjunctivity of composition } \} \\
 & (I \vartriangle eqsp)^{\cup} \circ pref \parallel pdly \;\sqcup\; (I \vartriangle eqsp)^{\cup} \circ pdly \parallel pref \\
= & \qquad \{ \text{ Definition 3.10b } \} \\
 & (I \vartriangle eqsp)^{\cup} \;\circ\; (pref \circ \ll) \vartriangle (pdly \circ \gg) \;\sqcup\; (I \vartriangle eqsp)^{\cup} \;\circ\; (pdly \circ \ll) \vartriangle (pref \circ \gg) \\
= & \qquad \{ \text{ Axiom 3.10d } \} \\
 & (pref \circ \ll \sqcap\; eqsp \circ pdly \circ \gg) \;\sqcup\; (pdly \circ \ll \sqcap\; eqsp \circ pref \circ \gg)
\end{array}$$

□

Next, the commutation properties of *Buff* with postcompose structures are considered. Because the definition of *Buff* is asymmetric, due to the left projection, we first look at a nice preliminary result:

Lemma 12.4

$$\begin{array}{ll}
 & Buff \circ I \parallel s^{\circ} \;=\; Buff \circ I \parallel (s^{\circ})^{>} \\
\Leftarrow & \\
 & s \bullet sm \;=\; sm \bullet s
\end{array}$$

Proof:
In this proof, we use a distribution property over cap. Actually, the distribution is in the axiomatisation of split, 3.10d:

$$\begin{array}{ll}
 & Buff \;\circ\; I \parallel s^{\circ} \\
= & \qquad \{ \text{ Lemma 12.3 } \} \\
 & ((pref \circ \ll \sqcap\; eqsp \circ pdly \circ \gg) \;\sqcup\; (pdly \circ \ll \sqcap\; eqsp \circ pref \circ \gg)) \;\circ\; I \parallel s^{\circ} \\
= & \qquad \{ \text{ cupjunctivity } \} \\
 & (pref \circ \ll \sqcap\; eqsp \circ pdly \circ \gg) \;\circ\; I \parallel s^{\circ} \;\sqcup\; (pdly \circ \ll \sqcap\; eqsp \circ pref \circ \gg) \;\circ\; I \parallel s^{\circ} \\
= & \qquad \{ \text{ Axiom 3.10d: distribution over cap } \} \\
 & (pref \circ \ll \sqcap\; eqsp \circ pdly \circ s^{\circ} \circ \gg) \;\sqcup\; (pdly \circ \ll \sqcap\; eqsp \circ pref \circ s^{\circ} \circ \gg) \\
= & \qquad \{ \text{ Theorem 10.18 } \} \\
 & (pref \circ \ll \sqcap\; pdly \circ eqsp \circ s^{\circ} \circ \gg) \;\sqcup\; (pdly \circ \ll \sqcap\; pref \circ eqsp \circ s^{\circ} \circ \gg)
\end{array}$$

$$
\begin{array}{ll}
= & \{ \text{ assumption } s \bullet sm = sm \bullet s \text{: Theorems 11.8, 11.7a and 11.7c } \} \\
& (pref \circ \ll \sqcap\ pdly \circ eqsp \circ (s^\circ)_\triangleright \circ \gg)\ \sqcup\ (pdly \circ \ll \sqcap\ pref \circ eqsp \circ (s^\circ)_\triangleright \circ \gg) \\
= & \{ \text{ same steps backwards } \} \\
& Buff \circ I \parallel (s^\circ)_\triangleright
\end{array}
$$

□

This proposition tells us that if a relation s acting on the request channel does not change the occurrence of messages, its actions may equally well be omitted. For the commutation with postcompose structures in general we have, analogously to Theorem 11.10:

Theorem 12.5 *Commutation with postcompose*

a.
$$r^\circ \circ Buff \ \sqsupseteq\ Buff \circ r^\circ \parallel s^\circ$$
$$\Leftarrow$$
$$r \bullet sm = sm \bullet r \ \wedge\ r \bullet wipe \supseteq wipe \bullet r \ \wedge\ s \bullet sm = sm \bullet s$$

b.
$$r^\circ \circ Buff \circ (r^\circ \parallel s^\circ)_\triangleright \ \sqsubseteq\ Buff \circ r^\circ \parallel s^\circ$$
$$\Leftarrow$$
$$r \bullet sm = sm \bullet r \ \wedge\ r \bullet wipe \bullet r_\triangleright \subseteq wipe \bullet r \ \wedge\ s \bullet sm = sm \bullet s$$

c.
$$r^\circ \circ Buff \circ (r^\circ \parallel s^\circ)_\triangleright \ =\ Buff \circ r^\circ \parallel s^\circ$$
$$\Leftarrow$$
$$r \bullet sm = sm \bullet r \ \wedge\ r \bullet wipe \bullet r_\triangleright = wipe \bullet r \ \wedge\ s \bullet sm = sm \bullet s$$

□

Now we derive some theorems for *Buff* in a loop. The loop-construction $(P \circ Buff)^\varpi$ which we will consider can be viewed as a buffered proc P. This construction can be very useful in case it is possible that input 'comes too quickly', or, alternatively stated, that P is too slow.
Four different inclusions are proved in Theorem 12.7. The conditions are on the request channel of proc P (the second channel of P). We will encounter a new interface *inf* in the condition of the third theorem, 12.7c. The condition of this property originally had the shape $A \circ pref \sqsubseteq pdly$ for some interface A. It prescribes that A does not contain chronicles that stop carrying messages at some moment. For a contradiction consider chronicle $f \in A$ and assume that after moment t there are no messages on f anymore. We extend this chronicle f after moment t with some messages to the chronicle f^+. Then clearly $f \langle A \circ pref \rangle f^+$, but $f \langle pdly \rangle f^+$ is *not* valid. We conclude that A only contains chronicles that never stop carrying messages (if A contains chronicles at all). The greatest monotype A satisfying the inequality $A \circ pref \sqsubseteq pdly$ is interface $pdly/pref \sqcap I$; this interface can be interpreted as the set of chronicles carrying messages for ever.

For reasons of elegance the following definition is taken:

Definition 12.6 *Infinite chronicles*
inf is an interface such that for all interface A

$$\begin{array}{ll} & A \sqsubseteq \mathit{inf} \\ \equiv & \\ & A \circ \mathit{pref} \sqsubseteq \mathit{pdly} \end{array}$$

□

We prefer Definition 12.6 over the explicit definition $\mathit{pdly}/\mathit{pref} \sqcap I$, since the use of the latter often necessitates an additional step to eliminate the factor. The four inequalities we want to prove are:

Theorem 12.7 *Buffer in loop*

a. $(P \circ \mathit{Buff})^{\varpi} \sqsubseteq \ll \circ P \circ \mathit{pref}$

b. $(P \circ \mathit{Buff} \circ I \parallel Q)^{\varpi} \sqsupseteq \ll \circ P \circ \mathit{pref} \Leftarrow \gg \circ P \sqsubseteq (\mathit{pdly} \circ \mathit{eqsp} \circ Q)^{\cup}$

c. $(P \circ \mathit{Buff})^{\varpi} \sqsubseteq \ll \circ P \circ \mathit{pdly} \Leftarrow \mathit{pdly} \circ \mathit{eqsp} \circ \gg \circ P \sqcap I \sqsubseteq \mathit{inf}$

d. $(P \circ \mathit{Buff} \circ I \parallel Q)^{\varpi} \sqsupseteq \ll \circ P \circ \mathit{pdly} \Leftarrow \gg \circ P \sqsubseteq (\mathit{pref} \circ \mathit{eqsp} \circ Q)^{\cup}$

Proof:
The first theorem is proved by a straightforward calculation. It expresses that any input of the proc P (inside the loop) is a prefix of the input of the buffered proc $(P \circ \mathit{Buff})^{\varpi}$:

$$\begin{array}{ll} & (P \circ \mathit{Buff})^{\varpi} \\ = & \quad \{ \text{ Lemma 12.3 } \} \\ & (P \circ ((\mathit{pref} \circ \ll \sqcap \mathit{eqsp} \circ \mathit{pdly} \circ \gg) \sqcup (\mathit{pdly} \circ \ll \sqcap \mathit{eqsp} \circ \mathit{pref} \circ \gg)))^{\varpi} \\ \sqsubseteq & \quad \{ \text{ monotonicity } \} \\ & (P \circ (\mathit{pref} \circ \ll \sqcup \mathit{pdly} \circ \ll))^{\varpi} \\ = & \quad \{ \text{ Property 10.16b: monotonicity } \} \\ & (P \circ \mathit{pref} \circ \ll)^{\varpi} \\ = & \quad \{ \text{ Definitions 3.17 and 3.14 } \} \\ & \ll \circ (P \circ \mathit{pref} \circ \ll \circ I \parallel \gg \sqcap \gg) \circ I \vartriangle \top\top \\ \sqsubseteq & \quad \{ \text{ Parallel computation 3.32c and monotonicity } \} \\ & \ll \circ P \circ \mathit{pref} \circ \ll \circ I \vartriangle \top\top \\ = & \quad \{ \text{ Split computation 3.32a } \} \\ & \ll \circ P \circ \mathit{pref} \end{array}$$

The second theorem gives a sufficient condition to ensure that *all* prefixes of the input (including the complete input) of the buffered proc $(P \circ \mathit{Buff})^{\varpi}$ can be consumed by P:

$$\begin{array}{ll} & (P \circ \mathit{Buff} \circ I \parallel Q)^{\varpi} \\ \sqsupseteq & \quad \{ \text{ Definition } \mathit{Buff}\text{; monotonicity } \} \\ & (P \circ \ll \circ \mathit{synp} \circ \mathit{pref} \parallel \mathit{pdly} \circ I \parallel Q)^{\varpi} \end{array}$$

$$
\begin{array}{ll}
= & \{ \text{ Parallel-parallel fusion 3.12b; Theorem 3.18b } \} \\
& (P \circ \ll \circ \mathit{synp} \circ I \parallel (\mathit{pdly} \circ Q))^{\varpi} \circ \mathit{pref} \\
= & \{ \text{ Definition 3.17 } \} \\
& \ll \circ (P \circ \ll \circ \mathit{synp} \circ I \parallel (\mathit{pdly} \circ Q) \sqcap \top\!\!\top \parallel I) \circ \ll^{\cup} \circ \mathit{pref} \\
= & \{ \text{ swap property; } \mathit{synp} \text{ is an interface } \} \\
& \ll \circ (P \circ \ll \circ \mathit{synp} \sqcap \top\!\!\top \parallel (\mathit{pdly} \circ Q)^{\cup}) \circ \mathit{synp} \circ \ll^{\cup} \circ \mathit{pref} \\
= & \{ \text{ } \mathit{synp} \circ \ll^{\cup} = (\ll \circ \mathit{synp})^{\cup} = I \vartriangle \mathit{eqsp} \text{ is an injection: Theorem 3.35b } \} \\
& \ll \circ (P \sqcap \top\!\!\top \parallel (\mathit{pdly} \circ Q)^{\cup} \circ I \vartriangle \mathit{eqsp}) \circ \mathit{pref} \\
= & \{ \text{ Parallel-split fusion 3.12a } \} \\
& \ll \circ (P \sqcap \top\!\!\top \vartriangle ((\mathit{pdly} \circ Q)^{\cup} \circ \mathit{eqsp})) \circ \mathit{pref} \\
= & \{ \text{ } \mathit{eqsp} = \mathit{eqsp}^{\cup} \text{: reverse } \} \\
& \ll \circ (P \sqcap \top\!\!\top \vartriangle (\mathit{eqsp} \circ \mathit{pdly} \circ Q)^{\cup}) \circ \mathit{pref} \\
= & \{ \text{ the condition equivales } I \parallel I \circ P \sqsubseteq \top\!\!\top \vartriangle (\mathit{eqsp} \circ \mathit{pdly} \circ Q)^{\cup} \} \\
& \ll \circ P \circ \mathit{pref}
\end{array}
$$

Proving the third theorem requires some preliminary work. We calculate, for all procs R:

$$
\begin{array}{ll}
& \mathit{Buff} \sqcap R \circ \gg \sqsubseteq \mathit{pdly} \circ \ll \\
= & \{ \text{ Lemma 12.3 } \} \\
& ((\mathit{pdly} \circ \ll \sqcap \mathit{eqsp} \circ \mathit{pref} \circ \gg) \sqcup (\mathit{pref} \circ \ll \sqcap \mathit{eqsp} \circ \mathit{pdly} \circ \gg)) \sqcap R \circ \gg \sqsubseteq \mathit{pdly} \circ \ll \\
= & \{ \text{ } \mathit{pdly} \circ \ll \sqcap \mathit{eqsp} \circ \mathit{pref} \circ \gg \sqcap R \circ \gg \sqsubseteq \mathit{pdly} \circ \ll \text{ } \} \\
& \mathit{pref} \circ \ll \sqcap \mathit{eqsp} \circ \mathit{pdly} \circ \gg \sqcap R \circ \gg \sqsubseteq \mathit{pdly} \circ \ll \\
= & \{ \ll \text{ is a total function } \} \\
& (\mathit{pref} \circ \ll \sqcap \mathit{eqsp} \circ \mathit{pdly} \circ \gg \sqcap R \circ \gg) \circ \ll^{\cup} \sqsubseteq \mathit{pdly} \\
= & \{ \ll \text{ and } \gg \text{ are functions: Theorem 3.35a and 3.35b } \} \\
& \mathit{pref} \sqcap (\mathit{eqsp} \circ \mathit{pdly} \sqcap R) \circ \gg \circ \ll^{\cup} \sqsubseteq \mathit{pdly} \\
= & \{ \gg \circ \ll^{\cup} = \top\!\!\top \} \\
& \mathit{pref} \sqcap (\mathit{eqsp} \circ \mathit{pdly} \sqcap R) \circ \top\!\!\top \sqsubseteq \mathit{pdly} \\
= & \{ \text{ Theorem 3.30a; } (P \sqcap Q)^{<} = P \circ Q^{\cup} \sqcap I \} \\
& (\mathit{eqsp} \circ \mathit{pdly} \circ R^{\cup} \sqcap I) \circ \mathit{pref} \sqsubseteq \mathit{pdly} \\
= & \{ \text{ } \mathit{eqsp} \circ \mathit{pdly} \circ R^{\cup} \sqcap I \text{ is an interface: Definition 12.6 } \} \\
& \mathit{eqsp} \circ \mathit{pdly} \circ R^{\cup} \sqcap I \sqsubseteq \mathit{inf}
\end{array}
$$

This lemma enables us to give a proof of Theorem 12.7c using only one implication:

$$
\begin{array}{ll}
& (P \circ \mathit{Buff})^{\varpi} \sqsubseteq \ll \circ P \circ \mathit{pdly} \\
= & \{ \text{ Definition 3.17 } \} \\
& \ll \circ (P \circ \mathit{Buff} \sqcap \top\!\!\top \parallel I) \circ \ll^{\cup} \sqsubseteq \ll \circ P \circ \mathit{pdly} \\
= & \{ \ll \text{ is a total function } \} \\
& P \circ \mathit{Buff} \sqcap \top\!\!\top \parallel I \sqsubseteq \ll^{\cup} \circ \ll \circ P \circ \mathit{pdly} \circ \ll \\
\Leftarrow & \{ \text{ Dedekind } \} \\
& P \sqcap \top\!\!\top \parallel I \circ \mathit{Buff}^{\cup} \sqsubseteq \ll^{\cup} \circ \ll \circ P \\
& \quad \wedge \quad \mathit{Buff} \sqcap P^{\cup} \circ \top\!\!\top \parallel I \sqsubseteq \mathit{pdly} \circ \ll \\
= & \{ \text{ } I \parallel I \sqsubseteq \ll^{\cup} \circ \ll \text{ } \} \\
& \mathit{Buff} \sqcap P^{\cup} \circ \top\!\!\top \parallel I \sqsubseteq \mathit{pdly} \circ \ll
\end{array}
$$

$$
\begin{array}{ll}
= & \{\ \top\top \parallel I = \gg^{\cup} \circ \gg;\ \text{reverse}\ \} \\
& \mathit{Buff} \sqcap (\gg \circ P)^{\cup} \circ \gg \sqsubseteq \mathit{pdly} \circ \ll \\
= & \{\ \text{result above}\ \} \\
& \mathit{eqsp} \circ \mathit{pdly} \circ \gg \circ P \sqcap I \sqsubseteq \mathit{inf}
\end{array}
$$

The proof of Theorem 12.7d is a variation on the proof of Theorem 12.7b, and therefore omitted

□

A corollary of Theorem 12.7b is:

$$(P \circ \mathit{Buff})^{\varpi} \sqsupseteq \ll \circ P \circ \mathit{pref} \ \Leftarrow\ \gg \circ P \sqsubseteq (\mathit{pdly} \circ \mathit{eqsp})^{\cup}$$

obtained by instantiating Q to the identity. Now consider, as an illustration, the following calculation for total Q:

$$
\begin{array}{ll}
& (P \circ \mathit{Buff} \circ I \parallel Q)^{\varpi} \sqsupseteq \ll \circ P \circ \mathit{pref} \\
= & \{\ \text{Theorem 3.18c; } Q \text{ is total: Parallel computation 3.32c}\ \} \\
& (I \parallel Q \circ P \circ \mathit{Buff})^{\varpi} \sqsupseteq \ll \circ I \parallel Q \circ P \circ \mathit{pref} \\
\Leftarrow & \{\ \text{above corollary}\ \} \\
& \gg \circ I \parallel Q \circ P \sqsubseteq (\mathit{pdly} \circ \mathit{eqsp})^{\cup} \\
= & \{\ \text{Parallel computation 3.32d}\ \} \\
& Q \circ \gg \circ P \sqsubseteq (\mathit{pdly} \circ \mathit{eqsp})^{\cup}
\end{array}
$$

Compare the condition in the last line with the condition of Theorem 12.7b: the condition derived in the above calculation is (for nondeterministic procs) stronger than the condition of Theorem 12.7b. So the calculator is well-adviced to move, before applying Theorem 12.7b, as many components as possible from the output side of the loop body to the input side, thereby weakening the condition. A similar remark holds for Theorem 12.7d. Finally, the weak causality of the buffer is derived. Unlike the problems we had in proving weak causality of *Sync*, the weak causality of *Buff* is obtained straightforwardly:

Theorem 12.8 *Weak causality of Buff*

- $$\mathit{Buff} \in I' \overset{w}{\rightsquigarrow} I' \parallel I$$

Proof:
Surjectivity is already stated in Theorem 12.2a, so we concentrate on the weak causality, totality and primed typing. Primed typing of *Sync* implies the condition of Axiom 9.8q on *Sync*: *detar.Sync*. So:

$$
\begin{array}{ll}
& \mathit{Buff} \in I' \overset{w}{\rightsquigarrow} I' \parallel I \\
= & \{\ \text{Definition 12.1}\ \} \\
& \ll \circ \mathit{Sync} \in I' \overset{w}{\rightsquigarrow} I' \parallel I \\
\Leftarrow & \{\ \ll\!> = I \parallel I \sqsupseteq \mathit{synp} = \mathit{Sync}\!<\!:\ \text{Axiom 9.8q}\ \} \\
& \ll \in I' \overset{w}{\rightsquigarrow} I' \parallel I \ \wedge\ \mathit{Sync} \in (I \parallel I)' \overset{w}{\rightsquigarrow} (I \parallel I)' \\
= & \{\ \text{Axiom 9.8c and primed typing; Property 11.11}\ \} \\
& \mathit{True}
\end{array}
$$

□

The second step in the proof above, where we derived the primed typing of $\ll \circ Sync$ from its components, is one of the preservation rules for primed typing as mentioned in Section 7.3. The kind of proof as given above will occur often: (weak) causality, totality and primed typing are proved at once. This is due to the requirements for obtaining (weak) causality (for example Axiom 9.8q) where we often use totality and primed typing information.
Sections 12.1 and 12.2 defined and investigated a lazy buffer; lazy in the sense that the buffer only gives output when there are corresponding requests. The following section defines an eager buffer which takes the initiative in sending the first message. We will take a more *axiomatic* approach to introduce new procs: no exact definitions are given; only the need for several assumptions (axioms) is motivated.

12.3 Hot buffer

In Section 12.1 we introduced a buffer. Theorem 12.7 lists several conditional assertions of a buffered proc P. In particular, the condition of Theorem 12.7c describes that the trigger channel of P carries more signals than the input channel. This was done to guarantee the progress of the complete buffered proc.
Until now, the progress was the responsibility of the proc P. In the extremal case $Buff^\sigma$, where there is no proc present which takes the responsibility, the proc might simply refuse to go on at some moment: $Buff^\sigma = pref$. Another possibility is to give the buffer the responsibility of progress. This buffer is called a *hot buffer*. We define it as the usual buffer *Buff* preceded by some proc *click* for which we have to find properties such that when the hot buffer *Hbuff* is put in a feedback it establishes a possible delay. The hot buffer is defined as follows for some proc *click*:

Definition 12.9 *Hot buffer*

$$Hbuff \triangleq Buff \circ I \parallel click$$

□

The result should be a buffer which always outputs (at least) its first input message, and after that acts as the normal buffer.
In the model, *click* is characterised as follows;

$$f \langle click \rangle g$$
$$\equiv$$
$$\forall(\, t \;::\; \#(f \bullet (< \bullet t)_\prec) \;\leqslant\; 1 + \#(g \bullet (< \bullet t)_\prec)\,) \;\wedge\; \#f \;=\; 1 + \#g$$

In words: the output is at most one message ahead (regardless of the actual contents of the messages), and the total number of output messages is one more (1+) than the number of input messages (where $1 + \omega = \omega$). Putting the hot buffer in a loop, we get progress 'for free': $Hbuff^\sigma = pdly$. This is the main property for *Hbuff* which will result in axioms on *click*.

First, Theorem 12.7 is adapted to the hot buffer:

Theorem 12.10 *Hot-buffered procs*

a. $(P \circ Hbuff)^{\varpi} \sqsubseteq \ll \circ P \circ pref$

b. $(P \circ Hbuff \circ I \parallel Q)^{\varpi} \sqsupseteq \ll \circ P \circ pref \Leftarrow \gg \circ P \sqsubseteq (click \circ Q)^{\cup}$

c. $(P \circ Hbuff)^{\varpi} \sqsubseteq \ll \circ P \circ pdly \Leftarrow click \circ \gg \circ P \sqcap I \sqsubseteq inf$

d. $(P \circ Hbuff \circ I \parallel Q)^{\varpi} \sqsupseteq \ll \circ P \circ pdly \Leftarrow \gg \circ P \sqsubseteq (pref \circ click \circ Q)^{\cup}$

□

This follows straightforwardly from the definition of *Hbuff* and Theorem 12.7 if we impose the following reasonable axioms on *click*:

Axiom 12.11 *Click*

a. $eqsp \circ click = click = click \circ eqsp$

b. $pdly \circ click = click = click \circ pdly$

□

That is, *click* is insensitive to contents of messages and possible delays. To get the key property $Hbuff^{\sigma} = pdly$, which demonstrates the progress obtained by *Hbuff*, we need more assumptions. They are derived as follows:

Theorem 12.12 *Key property*

$$Hbuff^{\sigma} = pdly$$

Proof:

$$\begin{array}{ll} & Hbuff^{\sigma} = pdly \\ = & \quad \{ \text{ Theorem 3.18a } \} \\ & (I \triangle I \circ Hbuff)^{\varpi} = pdly \\ = & \quad \{ \text{ plat calculus } \} \\ & (I \triangle I \circ Hbuff)^{\varpi} \sqsupseteq pdly \ \wedge \ (I \triangle I \circ Hbuff)^{\varpi} \sqsubseteq pdly \\ \Leftarrow & \quad \{ \text{ Theorem 12.10: } P := I \triangle I \text{ and } Q := I \text{; Split computation 3.32 } \} \\ & I \sqsubseteq (pref \circ click)^{\cup} \ \wedge \ click \sqcap I \sqsubseteq inf \\ = & \quad \{ \text{ reverse } \} \\ & I \sqsubseteq pref \circ click \ \wedge \ click \sqcap I \sqsubseteq inf \\ = & \quad \{ \text{ see below } \} \\ & True \end{array}$$

□

The last proof step gives us additional (necessary) assumptions for *click*. Later, yet another

assumption is needed; we record it already now:

Axiom 12.13 *Click*

a. $I \sqsubseteq pref \circ click$

b. $click \sqcap I \sqsubseteq inf$

c. $click \circ pref \sqsubseteq pref \circ click$

□

The first axiom says that the proc *click* is able to extend (in the sense of $pref^{\cup}$) any input chronicle with messages. It does not express that 'at the end' the total number of outputs is exactly one more than the total number of inputs. The second assumption is the most important one: it states that the interface $click \sqcap I$ represents a set of chronicles carrying messages for ever. The third one states a weak commutation property needed in the sequel. One of the important properties is the weak causality of *click*. This might seem strange for a proc that is able to produce one message ahead. But a possible implementation of *click* might be: copy all incoming messages possibly delayed to the output and insert an arbitrary extra message somewhere. We can abstract from the exact contents by sequentially composing it with the weakly causal proc *eqsp*. Then we get the proc *click*, which is weakly causal:

Axiom 12.14 *Weak causality of click*

$$\forall(\, X \;:\; X \in \mathcal{D} \;:\; X \circ click \in X \overset{w}{\rightsquigarrow} I\,)$$

□

This assumption makes it possible to conclude weak causality of *click* and *Hbuff*. Observe that $X \circ click$ obeys the inclusion $X \circ click \circ eqar \circ click^{\cup} \circ X \sqsubseteq eqar$, which is *not* satisfied by *click* because the latter proc absorbs *eqsp*, Axiom 12.11a, and the proc *eqsp* is able to destroy the arity information completely.

Theorem 12.15 *Weak causality of click and Hbuff*

a. *wcausal.click*

- b. $Hbuff \;\in\; I' \overset{w}{\rightsquigarrow} I' \parallel I$

Proof:
Weak causality of *click* follows from Axioms 12.14 and 9.8o and Lemma 7.8a. The same axioms are used to prove 12.15b:

$$\begin{array}{ll} & Hbuff \;\in\; I' \overset{w}{\rightsquigarrow} I' \parallel I \\ = & \quad \{\ \text{Definition 12.9}\ \} \\ & Buff \circ I \parallel click \;\in\; I' \overset{w}{\rightsquigarrow} I' \parallel I \\ = & \quad \{\ \text{Lemma 7.8a}\ \} \\ & Buff \;\circ\; I \;\parallel\; (\sqcup(\, X \;::\; X\,) \circ click) \;\in\; I' \overset{w}{\rightsquigarrow} I' \parallel I \end{array}$$

$$
\begin{array}{ll}
= & \{ \text{ cupjunctivity } \} \\
& \sqcup(\, X \ :: \ \mathit{Buff} \circ I \parallel (X \circ \mathit{click})\,) \ \in \ I' \ \overset{w}{\rightsquigarrow} \ I' \parallel I \\
\Leftarrow & \{ \text{ Axiom 9.8o, domains and primed typing } \} \\
& \forall(\, X \ :: \ \mathit{Buff} \circ I \parallel (X \circ \mathit{click}) \ \in \ I' \ \overset{w}{\rightsquigarrow} \ I' \parallel I\,)
\end{array}
$$

We are heading for an application of Axiom 9.8q. To guarantee a fixed arity of the messages generated by *click*, the X is placed after this proc. For arbitrary proc P:

$$
\begin{array}{ll}
& P \ \in \ I' \parallel X \ \sim \ I' \parallel I \\
= & \{ \text{ calculation like Lemma 7.10 } \} \\
& P \ \sqsubseteq \ \mathit{eqar} \parallel (X \circ \top\!\!\top) \\
\Rightarrow & \{ \text{ relational calculus applying Lemma 7.8b } \} \\
& \mathit{detar}.P
\end{array}
$$

So if $P \ \in \ I' \parallel X \ \sim \ I' \parallel I$, it satisfies the important condition of determination of arities in Axiom 9.8q. This property is exploited in the first step of the following calculation: for all $X \in \mathcal{D}$,

$$
\begin{array}{ll}
& \mathit{Buff} \circ I \parallel (X \circ \mathit{click}) \ \in \ I' \ \overset{w}{\rightsquigarrow} \ I' \parallel I \\
\Leftarrow & \{ \text{ Axiom 9.8q } \} \\
& \mathit{Buff} \ \in \ I' \ \overset{w}{\rightsquigarrow} \ I' \parallel I \ \wedge \ I \parallel (X \circ \mathit{click}) \ \in \ I' \parallel X \ \overset{w}{\rightsquigarrow} \ I' \parallel I \\
\Leftarrow & \{ \text{ Theorem 12.8; Axiom 9.8r } \} \\
& I \in I' \ \overset{w}{\rightsquigarrow} I' \ \wedge \ X \circ \mathit{click} \in X \ \overset{w}{\rightsquigarrow} I \\
= & \{ \text{ Theorem 9.7b and primed typing; Axiom 12.14 } \} \\
& \mathit{True}
\end{array}
$$

□

For later use we prove one last lemma about the hot buffer:

Lemma 12.16

$$\mathit{Hbuff} \circ \gg^{\cup} \sqsubseteq \mathit{pref} \circ \mathit{click}$$

Proof:

$$
\begin{array}{ll}
& \mathit{Hbuff} \circ \gg^{\cup} \\
= & \{ \text{ Definition 12.9; Lemma 12.3 } \} \\
& ((\mathit{pref} \circ \ll \sqcap \ \mathit{eqsp} \circ \mathit{pdly} \circ \gg) \ \sqcup \ (\mathit{pdly} \circ \ll \sqcap \ \mathit{eqsp} \circ \mathit{pref} \circ \gg)) \ \circ \ I \parallel \mathit{click} \ \circ \ \gg^{\cup} \\
\sqsubseteq & \{ \text{ monotonicity; reversed Parallel computation 3.32d } \} \\
& (\mathit{eqsp} \circ \mathit{pdly} \circ \gg \sqcup \ \mathit{eqsp} \circ \mathit{pref} \circ \gg) \ \circ \ \gg^{\cup} \ \circ \ \mathit{click} \\
= & \{ \text{ Property 10.16b: monotonicity } \} \\
& \mathit{eqsp} \circ \mathit{pref} \circ \gg \circ \gg^{\cup} \circ \mathit{click} \\
\sqsubseteq & \{ \text{ Theorem 10.18; } \gg \text{ is a function } \} \\
& \mathit{pref} \circ \mathit{eqsp} \circ \mathit{click} \\
= & \{ \text{ Axiom 12.11a } \} \\
& \mathit{pref} \circ \mathit{click}
\end{array}
$$

□

This concludes the short discussion on hot buffers. The primitive proc *click* turns out to be useful in other definitions of buffers also, for example in the definition of a one-place buffer. This topic will be discussed in the next section.

12.4 n-place buffer

In Section 12.1 we discussed several other approaches to the process of buffering. In particular the buffer defined in Hoare [Hoa85] (and Baeten & Weijland [BW90]) differs significantly. There, the buffer is not triggered by some explicit request; it simply describes the I/O-behaviour of the messages only (observed by some observer).
In our calculus we can define such a proc also. We start with a one-place buffer and build buffers of arbitrary but finite length with it. A one-place buffer can store at most one message which can be sent to the output any time (if the next proc is able to receive it). A new message can only be sent to the one-place buffer if it is empty. Other I/O-behaviour is not described:

Definition 12.17 *n-place buffer*

a. $(1) \triangleq pref \sqcap (pref \circ click)^{\cup}$

b. $(n) \triangleq (1)^n$

□

In the model, (1) looks as follows:

$$\begin{array}{ll} & f\,\langle (1)\rangle\, g \\ \equiv & \\ & f\,\langle pref\rangle\, g \wedge \forall(\, t \;::\; \#(g \bullet (<\bullet t)_{\prec}) \;\leqslant\; 1 + \#(f \bullet (<\bullet t)_{\prec})\,) \end{array}$$

In words: the output is a prefix of the input with the restriction that the output lags at most one message behind. A number of properties follow readily from Definition 12.17. As an exercise we show the next theorem:

Theorem 12.18 *n-place buffer*

a. $(n) \circ (m) = (n+m)$

b. $I \sqsubseteq (n) \sqsubseteq pref$

c. $(n) \sqsubseteq (m) \;\Leftarrow\; n \leqslant m$

d. $(n) \in I' \rightsquigarrow I'$

e. $eqsp \circ (n) = (n) \circ eqsp$

Proof:
From the definition of P^n, Definitions 3.7a and 3.7b, it follows straightforwardly by induction that $P^n \circ P^m = P^{n+m}$. By Definition 12.17, this implies 12.18a. 12.18b is proved by induction over n. The basis is trivially met because of $P^0 = I$ and Property 10.16a. For the step we proceed as follows:

$$
\begin{array}{ll}
 & I \sqsubseteq (n+1) \sqsubseteq \mathit{pref} \\
= & \quad \{ \text{ Theorem 12.18a } \} \\
 & I \sqsubseteq (n) \circ (1) \sqsubseteq \mathit{pref} \\
\Leftarrow & \quad \{ \text{ monotonicity: Property 10.16f } \} \\
 & I \sqsubseteq (n) \sqsubseteq \mathit{pref} \ \wedge \ I \sqsubseteq (1) \sqsubseteq \mathit{pref} \\
= & \quad \{ \text{ Induction Hypothesis } \} \\
 & I \sqsubseteq (1) \sqsubseteq \mathit{pref} \\
= & \quad \{ \text{ Definition 12.17a } \} \\
 & I \ \sqsubseteq \ \mathit{pref} \sqcap (\mathit{pref} \circ \mathit{click})^{\cup} \ \sqsubseteq \ \mathit{pref} \\
= & \quad \{ \text{ plat calculus } \} \\
 & I \sqsubseteq \mathit{pref} \ \wedge \ I \sqsubseteq (\mathit{pref} \circ \mathit{click})^{\cup} \\
= & \quad \{ \text{ Property 10.16a; Axiom 12.13a and reverse } \} \\
 & \mathit{True}
\end{array}
$$

Theorem 12.18c follows by monotonicity from the first two properties. 12.18d is derived from the observation that $I \sqsubseteq (n) \sqsubseteq \mathit{pref} \sqsubseteq \mathit{eqar}$. Notice that reflexivity of (n) implies totality and surjectivity. Finally, 12.18e follows by induction from the fact that *click* absorbs *eqsp*, Axiom 12.11, and that *pref* commutes with *eqsp*, Theorem 10.18
□

Assuming the axiom $I = \mathit{pref}^{\cup} \sqcap \mathit{pref}$ we can derive that the n-place buffer (n) refines the proc $\mathit{pref} \sqcap (\mathit{pref} \circ \mathit{click}^n)^{\cup}$. A real problem is the *equality* of the procs (n) and $\mathit{pref} \sqcap (\mathit{pref} \circ \mathit{click}^n)^{\cup}$: it is unclear what assumptions we need.
The refinement $\sqcup(\, n \ :: \ (n)\,) \leq \mathit{pref}$ is valid; this is a direct consequence of Theorem 12.18b. Equality does *not* hold because it could be the case that *pref* cuts an infinite number of messages. If we had defined something like P^{ω}, a possible equality would have been $\mathit{pref} = (\omega)$.

Summarising the results

The procs *pdly*, *dly* and *pref* are axiomatised as follows:

Axiom 12.19

a. $I \sqcup dly \sqsubseteq pdly \sqsubseteq pref \sqsubseteq eqar$

b. $pdly \circ pdly \sqsubseteq pdly$

c. $dly \circ pdly \sqcup pdly \circ dly \sqsubseteq dly$

d. $pref \circ pref \sqsubseteq pref$

- e. $dly \circ dly \sqsupseteq dly$

f. $dly \circ pref = pref \circ dly$

g. $r^\circ \circ pdly = pdly \circ r^\circ$

h. $r^\circ \circ dly = dly \circ r^\circ$

i. $r^\circ \circ pref \sqsupseteq pref \circ r^\circ \Leftarrow r \bullet wipe \supseteq wipe \bullet r$

j. $r^\circ \circ pref \circ (r^\circ)_\triangleright \sqsubseteq pref \circ r^\circ \Leftarrow r \bullet wipe \bullet r_\triangleright \subseteq wipe \bullet r$

k. $r^\circ \circ pref \circ (r^\circ)_\triangleright = pref \circ r^\circ \Leftarrow r \bullet wipe \bullet r_\triangleright = wipe \bullet r$

The causality assumptions on these three procs are:

l. *wcausal.pdly*

m. *causal.dly*

n. *wcausal.pref*

In contrast to the procs *pdly*, *dly* and *pref*, the proc *Sync* is not axiomatised but defined: Definition 11.2. To get all the desired properties, several axioms have to be imposed:

o. $I = eqsp \sqcap pdly^{\cup} \circ pref$

- p. $\leqslant_{\#} \sqcup \geqslant_{\#} = \top\top$

And finally for the causality:

- q. $wcausal.Sync$

□

All the properties of Part IV which are not recorded in Axiom 12.19 are derivable. Therefore, they are referred to as *theorems* in the following part.

Part V

An application

Chapter 13

Instantiating the calculus

We strive for a general theory. The results of this theory can be instantiated to more specific theorems which are useful in practical applications.
The set of interfaces (types) is rather rich in the sense that any union of interfaces is again an interface; this is not a useful property in practical applications. Moreover, the choice for using an untyped calculus forced us to impose assumptions on relations and procs such as primed typing and preservation of arities. This chapter presents the set of interfaces that have a well-defined arity. This results in simple typing rules. Furthermore, typed versions of polymorphic procs are defined and investigated.

13.1 Typed rules

Starting off with a calculus of relations with well-defined (input and output) arities frees us from the need to impose polyfunctionality, primed typing and conditions like the preservation of arities. Formally, 'well-defined arity' for interface A is described by the inclusion $A \circ \top\top \circ A \sqsubseteq \mathit{eqar}$. Several axioms listed in Axiom 9.8 reduce to simple typing rules if we only use interfaces with a well-defined arity. We record a few typing rules needed in the next chapter:

Corollary 13.1 *Typing Rules*
For interfaces A, B, C and D with a well-defined arity:

a. $$P \circ Q \in A \overset{c}{\rightsquigarrow} B$$
$$\Leftarrow$$
$$(P \in A \overset{w}{\rightsquigarrow} C \ \wedge\ Q \in C \overset{c}{\rightsquigarrow} B) \ \vee\ (P \in A \overset{c}{\rightsquigarrow} C \ \wedge\ Q \in C \overset{w}{\rightsquigarrow} B)$$

b. $$P \circ Q \in A \overset{w}{\rightsquigarrow} B \ \Leftarrow\ P \in A \overset{w}{\rightsquigarrow} C \ \wedge\ Q \in C \overset{w}{\rightsquigarrow} B$$

c. $$P \vartriangle Q \ \in\ A \parallel B \ \overset{w}{\rightsquigarrow}\ C \ \Leftarrow\ P \in A \overset{w}{\rightsquigarrow} C \ \wedge\ Q \in B \overset{w}{\rightsquigarrow} C$$

d. $$P \parallel Q \ \in\ A \parallel C \ \overset{w}{\rightsquigarrow}\ B \parallel D \ \Leftarrow\ P \in A \overset{w}{\rightsquigarrow} B \ \wedge\ Q \in C \overset{w}{\rightsquigarrow} D$$

e. $P^\sigma \in A \overset{c}{\rightsquigarrow} B \;\Leftarrow\; P \in A \overset{c}{\rightsquigarrow} B \parallel A$

□

All symbols $\sim$ in Corollary 13.1 can be replaced by $\leftarrow$ to obtain the set of rules for deterministic procs.
The refinement order $\leq$ boils down to ordinary relational inclusion $\sqsubseteq$ if we work with typed (and causal) procs. Moreover, demonic composition $\natural$ equals angelic composition $\circ$.
The refinement order $\leq$ is preserved by the combining forms $\sqcup$, $\circ$, $\triangle$, $\parallel$ and $^\sigma$. In particular the rule for feedback looks nice. For interfaces A and B with a well-defined arity:

$$P^\sigma \leq Q^\sigma \;:\; A \overset{c}{\rightsquigarrow} B$$
$$\Leftarrow$$
$$P \leq Q \;:\; A \overset{c}{\rightsquigarrow} B \parallel A$$

This rule is invalid if causality is omitted.

13.2 Stream types

The discussion above suggests a subset of the procs which could be valuable with respect to implementations. A set of types containing only interfaces representing chronicles over *base types*—such as the naturals—might be useful. The procs which are typed with those interfaces form a nice subset of the procs considered in this thesis.

Definition 13.2 *Base type, stream type*
Identity a is called a *base type* whenever $\bot \neq a \subseteq sm$ and $unar.a$. For base type a, we define the interface I_a of streams over a by:

$$I_a \triangleq (a \cup (wipe \bullet a)_\prec)^\circ$$

□

In words: base type a is a non-empty subset of the real messages sm, and all elements of a have the same arity. Base types are closed under product $\times$, but not under cup $\cup$ or cap $\cap$. The interface I_a represents what most people call *the (finite and infinite) streams over set* a. To get the type I_a, base type a is closed by adding the no-message $(wipe \bullet a)_\prec$, which has the arity of a, and this extension is lifted with postcompose.
Due to the unique arity of base type a, the interface I_a also has a well-defined arity. Furthermore, products of the base types are almost preserved by the extension and lifting; in conjunction with the interface *synp* of synchronised pairs products *are* preserved:

Theorem 13.3 *Stream type*

a. $I_a \circ \top\top \circ I_a \sqsubseteq eqar$

b. $I_{a\times b} = I_a \parallel I_b \circ synp$

□

The proof is omitted, but preservation properties of postcompose feature prominently. Observe that Theorem 13.3b would not hold if base type a or b were allowed to contain no-messages: the inclusion $I_{a\times b} \sqsubseteq synp$ would have been invalid. This is one of the reasons to require that a base type does not contain no-messages.
It is easy now to define several typed versions of primitive procs and defined procs of the calculus. In fact, I_a itself is just the typed interface I. Because we only need a few typed procs in the next chapter, we confine ourselves to the following four definitions:

Definition 13.4 *Typed procs*

a. $\ll_{ab} \triangleq \ll \circ I_a \parallel I_b$

b. $pdly_a \triangleq pdly \circ I_a$

c. $dly_a \triangleq dly \circ I_a$

d. $Hbuff_{ab} \triangleq Hbuff \circ I_a \parallel I_b$

□

Useful properties of the generic versions carry over to the typed procs. Among those properties are totality, functionality and (weak) causality:

Theorem 13.5 *Typed procs*

a. $I_a \in I_a \overset{w}{\leftarrowtail} I_a$

b. $\ll_{ab} \in I_a \overset{w}{\leftarrowtail} I_a \parallel I_b$

c. $pdly_a \in I_a \overset{w}{\rightsquigarrow} I_a$

d. $dly_a \in I_a \overset{c}{\rightsquigarrow} I_a$

• e. $Hbuff_{ab} \in I_a \overset{w}{\rightsquigarrow} I_a \parallel I_b$

Proof:
Because non-empty interfaces of the form c° for identity c are closed, the proof of 13.5a boils down to showing that $I_a \neq \perp\!\!\!\perp$. This follows from Axiom 6.15a.
We only give the proof of 13.5e. The proof obligation has to be split into two conjuncts: $Hbuff_{ab} \in I \overset{w}{\rightsquigarrow} I_a \parallel I_b$ and $I_a \circ Hbuff_{ab} = Hbuff_{ab}$. First, the totality and weak causality are proved together. Because $I_a \parallel I_b$ is an interface, the arity-determination condition of Axiom 9.8q is fulfilled:

$$
\begin{array}{ll}
 & Hbuff_{ab} \in I \overset{w}{\rightsquigarrow} I_a \parallel I_b \\
= & \quad \{ \text{ Definition 13.4d } \} \\
 & Hbuff \circ I_a \parallel I_b \in I \overset{w}{\rightsquigarrow} I_a \parallel I_b \\
\Leftarrow & \quad \{ \ I_a \parallel I_b \text{ has well-defined arity, and } I \parallel I \sqsupseteq I_a \parallel I_b \text{: Axiom 9.8q } \}
\end{array}
$$

$$
\begin{array}{ll}
 & Hbuff \in I \overset{w}{\rightsquigarrow} I \parallel I \ \wedge\ I_a \parallel I_b \in I_a \parallel I_b \overset{w}{\rightsquigarrow} I_a \parallel I_b \\
\Leftarrow & \quad \{ \text{ Theorem 12.15b; Axiom 9.8r } \} \\
 & I_a \in I_a \overset{w}{\rightsquigarrow} I_a \ \wedge\ I_b \in I_b \overset{w}{\rightsquigarrow} I_b \\
= & \quad \{ \text{ Theorem 13.5a } \} \\
 & True
\end{array}
$$

The typing $I_a \circ Hbuff_{ab} = Hbuff_{ab}$ requires several properties about identities and the idempotent function *wipe*:

$$
\begin{array}{ll}
 & I_a \circ Hbuff_{ab} = Hbuff_{ab} \\
= & \quad \{ \text{ Definition 13.4d } \} \\
 & I_a \circ Hbuff \circ I_a \parallel I_b \ =\ Hbuff \circ I_a \parallel I_b \\
\Leftarrow & \quad \{ \text{ compose with } I \parallel (click \circ I_b) \text{: Definition 12.9 } \} \\
 & I_a \circ Buff \circ I_a \parallel I \ =\ Buff \circ I_a \parallel I \\
= & \quad \{ \text{ interfaces } \} \\
 & I_a \circ Buff \ \sqsupseteq\ Buff \circ I_a \parallel I \\
\Leftarrow & \quad \{ \text{ Definition 13.2: Theorem 12.5a with } s = \mathcal{I} \ \} \\
 & (a \cup (wipe \bullet a)^{\prec}) \bullet sm \ =\ sm \bullet (a \cup (wipe \bullet a)^{\prec}) \\
 & \quad \wedge\ (a \cup (wipe \bullet a)^{\prec}) \bullet wipe \ \supseteq\ wipe \bullet (a \cup (wipe \bullet a)^{\prec}) \\
= & \quad \{ \ a \cup (wipe \bullet a)^{\prec} \text{ and } sm \text{ are identities: Corollary 3.27b } \} \\
 & (a \cup (wipe \bullet a)^{\prec}) \bullet wipe \ \supseteq\ wipe \bullet (a \cup (wipe \bullet a)^{\prec}) \\
\Leftarrow & \quad \{ \text{ properties cup } \} \\
 & (wipe \bullet a)^{\prec} \bullet wipe \supseteq wipe \bullet a \ \wedge\ (wipe \bullet a)^{\prec} \bullet wipe \supseteq wipe \bullet (wipe \bullet a)^{\prec} \\
= & \quad \{ \text{ for identity } c\text{: } (r \bullet c)^{\prec} \bullet r \supseteq r \bullet c \ \} \\
 & (wipe \bullet a)^{\prec} \bullet wipe \supseteq wipe \bullet (wipe \bullet a)^{\prec} \\
= & \quad \{ \text{ for identity } c \text{ and idempotent } r\text{: } (r \bullet c)^{\prec} \bullet r \supseteq r \bullet (r \bullet c)^{\prec} \ \} \\
 & True
\end{array}
$$

□

After having made this small instantiation of our calculus, we can tackle a well-known communication protocol.

Chapter 14

The Alternating Bit Protocol

As an example of the calculus for the design of distributed algorithms we want to consider a protocol; more specifically, the protocol we will end up with is the Alternating Bit Protocol. We do not claim to be the first in doing so: the *ABP* is one of the most extensively investigated protocols. In fact, the *ABP* may be considered a *test case* for calculi developed for the design of distributed algorithms, see for example Chandy & Misra [CM89], Möller [Möl94] or Vaandrager [Vaa90].
The protocol, which can be traced back to Bartlett *et al.* [BSW69], establishes the connection between the physical activities of transmitting raw bits over communication channels and the higher level activities of routing packets of information in a network. In terms of the ISO/OSI reference model, see Zimmerman [Zim80] or Tanenbaum [Tan81], the protocol transforms the raw transmission facilities of the *physical layer* into the reliable transmission facilities of the *datalink layer*.
In this chapter, we will demonstrate our calculus by deriving the Alternating Bit Protocol as a refinement of the delay *dly*. Given a faulty medium *flch*, the protocol is obtained in three phases. In Section 14.1, a feedback circuit is considered which contains a buffer and two media. It mainly prescribes the topology of the protocol we are heading for. The first calculations will remove the feedback circuit while imposing conditions on the two media. The result is that the feedback circuit refines *dly*. Then, we turn our attention in Section 14.2 to one particular medium with slightly different conditions, and show that it implements the two media of Section 14.1. Finally, in Section 14.3, this particular medium is instantiated to the faulty channel *flch* with adequate preprocessing and postprocessing. This immediately results in definitions for a sender S and a receiver R realising the *ABP*.
The further we proceed with the development of the protocol, the more typed procs will occur. We want to emphasise that the type information of a typed proc, given as a subscript, will *never* be omitted. In category theory, for example, type information is often left implicit, because the context gives enough information to deduce the type (objects) of the functions (arrows). In our system, it is impossible to omit type information because that would mix up the truly polymorphic procs (like *dly*) and the typed procs (like dly_a for some a).

14.1 The Basic Network

Consider the topology of the Alternating Bit Protocol in Figure 14.1. The channels flch_x and flch_y are faulty channels, and the task is to find a sender S, a receiver R and base types x and y such that the network refines a fault-free channel.

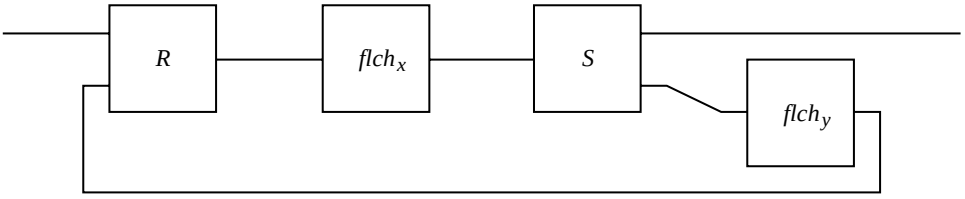

Figure 14.1: The ABP topology

As a proc, the topology of the ABP is defined with a loop:

Definition 14.1 *Alternating Bit Protocol*

$$ABP_{xy}.(R,S) \triangleq (R \circ \mathit{flch}_x \circ S \circ I \parallel \mathit{flch}_y)^{\varpi}$$

□

Instead of trying to solve the refinement problem of the ABP at once, we like to start with a simpler network as shown in Figure 14.2 and find conditions on channels ch_1 and ch_2 such that this Basic Network implements a reliable channel. Afterwards, definitions of R and S can be derived, given a faulty channel flch.
The network in Figure 14.2 is a hot buffer Hbuff placed in a feedback loop. The loop contains two media ch_1 and ch_2 which transmit messages from the hot buffer to the output, or transmit messages back to (the request channel of) the hot buffer.

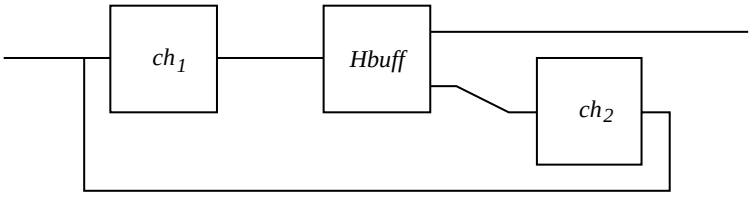

Figure 14.2: Basic Network

In terms of the calculus, this network translates to the following expression:

Definition 14.2 *Basic Network*

$$BN.(ch_1, ch_2) \triangleq (ch_1 \circ \mathit{Hbuff} \circ I \parallel ch_2)^{\sigma}$$

□

The problem is that the media ch_1 and ch_2 might be unreliable: it is not guaranteed that, under all circumstances, a message which is sent to one of the channels will actually be

transmitted. We are interested in the following question: under what conditions on ch_1 and ch_2 is (a typed version of) $BN.(ch_1, ch_2)$ a refinement of (a typed version of) dly, i.e., a reliable channel?
By Definition 7.31, refinement is divided into two parts: an inclusion and a domain property. We first concentrate on the inclusion $BN.(ch_1, ch_2) \sqsubseteq dly$, which is a combination of safety and liveness properties, see Lamport [Lam77]. The safety part contains the property that any message that is produced by the network has really been put in on an earlier moment, and there is no reordering. The liveness part includes the property that every message which is sent to the network will eventually be transmitted.
The proc dly should be a possible implementation for the media ch_1 and ch_2: without additional assumptions on the channel dly the inclusion $BN.(dly, dly) \sqsubseteq dly$ has to be valid:

Lemma 14.3

$$BN.(dly, dly) \sqsubseteq dly$$

Proof:
We are heading for an appeal to Theorem 12.10c:

$$\begin{array}{ll} & BN.(dly, dly) \sqsubseteq dly \\ = & \quad \{ \text{ Definition 14.2 } \} \\ & (dly \circ Hbuff \circ I \parallel dly)^\sigma \sqsubseteq dly \\ = & \quad \{ \text{ Theorems 3.18a and 3.18c } \} \\ & (I \triangle dly \circ dly \circ Hbuff)^\varpi \sqsubseteq dly \\ = & \quad \{ \text{ Split computation 3.32a; } dly = dly \circ pdly \ \} \\ & (I \triangle dly \circ dly \circ Hbuff)^\varpi \sqsubseteq \ll \circ I \triangle dly \circ dly \circ pdly \\ \Leftarrow & \quad \{ \text{ Theorem 12.10c } \} \\ & click \circ \gg \circ I \triangle dly \circ dly \sqcap I \sqsubseteq inf \\ \Leftarrow & \quad \{ \text{ Split computation 3.32b; } dly \circ dly \sqsubseteq dly \ \} \\ & click \circ dly \sqcap I \sqsubseteq inf \\ \Leftarrow & \quad \{ \text{ Axiom 12.19a: } dly \sqsubseteq pdly \ \} \\ & click \circ pdly \sqcap I \sqsubseteq inf \\ = & \quad \{ \text{ Axioms 12.11b and 12.13b } \} \\ & True \end{array}$$

□

To get an equality in Lemma 14.3 we need axioms like Axiom 12.19e. We do not, however, need an equality; the inequality suffices.
The lemma is valuable: it shows how we can transform a delay into a feedback structure containing a hot buffer, and we know more about the particular inclusion $P^\sigma \sqsubseteq Q^\sigma$ than about the more general inclusion $P^\sigma \sqsubseteq Q$.

We state the following conditions about the media ch_1 and ch_2. Later, more conditions such as causality will be added to obtain totality of the Basic Network:

Condition 14.4 *One-place channel*

a. $ch_1 \sqsubseteq pref \circ eqsp$

b. $ch_1 \sqcap (1) \circ eqsp \sqsubseteq dly$

c. $ch_2 \sqsubseteq pref \circ eqsp$

d. $ch_2 \sqcap (1) \circ eqsp \sqsubseteq dly \circ eqsp$

□

Condition 14.4a mainly says that the number of output messages of ch_1 does not exceed the number of input messages. Abstracting from timing, it directly results in the property $pdly^{\cup} \circ ch_1 \sqsubseteq \leqslant_{\#}$.
To explain Condition 14.4b, consider just a single message m which is sent to ch_1. According to Condition 14.4a we can only expect no output or a single output; the conjunct $(1) \circ eqsp$ agrees with this behaviour. Therefore, the inclusion in dly prescribes that the channel *has to transmit the message m eventually once*; no output is not allowed. This is a *liveness property*. The inclusion in dly also prescribes that *no other message is transmitted*. This is a *safety property*. When message m has been transmitted, a new message can be sent to the channel immediately and the scenario is repeated.
In short, Condition 14.4b states that a message which is sent to the medium ch_1 eventually is transmitted *as long as no other message is sent to the channel*. Therefore, channel ch_1 is called a one-place channel. Condition 14.4d is a weaker version of 14.4b: it does not require that the same message comes out; only the *timing* of the message is important.
These are stringent conditions, but far easier to check than an expression containing a feedback loop. Moreover, using a less reliable channel, the conditions 14.4 can be met by, for example, adding procs which *tag* and *repeatedly send* messages. Then, conditions like *weak fairness* enter the calculations, see Francez [Fra86]. Condition 14.4 takes care of the first part of the required inclusion for refinement:

Lemma 14.5

$$BN.(ch_1, ch_2) \sqsubseteq dly$$

Proof:

$$\begin{array}{ll} & BN.(ch_1, ch_2) \sqsubseteq dly \\ = & \quad \{ \text{ Definition 14.2 } \} \\ & (ch_1 \circ Hbuff \circ I \,\|\, ch_2)^{\sigma} \sqsubseteq dly \\ \Leftarrow & \quad \{ \text{ Lemma 14.3 } \} \\ & (ch_1 \circ Hbuff \circ I \,\|\, ch_2)^{\sigma} \sqsubseteq (dly \circ Hbuff \circ I \,\|\, dly)^{\sigma} \\ \Leftarrow & \quad \{ \text{ Definition 3.14: } P^{\sigma} \sqsubseteq Q^{\sigma} \Leftarrow P \sqcap \gg \sqsubseteq Q \ \} \\ & ch_1 \circ Hbuff \circ I \,\|\, ch_2 \sqcap \gg \sqsubseteq dly \circ Hbuff \circ I \,\|\, dly \end{array}$$

$$
\begin{array}{ll}
\Leftarrow & \quad \{ \text{ Dedekind 3.21 } \} \\
& ch_1 \sqcap \gg \circ (Hbuff \circ I \parallel ch_2)^{\cup} \sqsubseteq dly \\
& \quad \wedge \;\; Hbuff \circ I \parallel ch_2 \sqcap ch_1{}^{\cup} \circ \gg \sqsubseteq Hbuff \circ I \parallel dly \\
= & \quad \{ \; Hbuff = Hbuff \circ I \parallel eqsp \text{ and } eqsp \circ dly = dly \circ eqsp \; \} \\
& ch_1 \sqcap \gg \circ (Hbuff \circ I \parallel ch_2)^{\cup} \sqsubseteq dly \\
& \quad \wedge \;\; Hbuff \circ I \parallel ch_2 \sqcap ch_1{}^{\cup} \circ \gg \sqsubseteq Hbuff \circ I \parallel (dly \circ eqsp) \\
\Leftarrow & \quad \{ \text{ Dedekind 3.21; reverse } \} \\
& ch_1 \sqcap \gg \circ (Hbuff \circ I \parallel ch_2)^{\cup} \sqsubseteq dly \\
& \quad \wedge \;\; I \parallel ch_2 \sqcap (ch_1 \circ Hbuff)^{\cup} \circ \gg \sqsubseteq I \parallel (dly \circ eqsp)
\end{array}
$$

We derived two conjuncts which have a great similarity. The second conjunct contains several parallel compositions which do not occur in the first one. Relational calculations show us that these parallel compositions can be eliminated:

$$I \parallel P \sqcap Q \circ \gg \sqsubseteq I \parallel R \;\; \equiv \;\; P \sqcap \gg \circ Q \sqsubseteq R$$

This property can be derived using the fact that the projection is a function, total on the interface $I \parallel I$, Theorem 3.40f. Applying this property to the second conjunct derived above, we continue the calculation:

$$
\begin{array}{ll}
& ch_1 \sqcap \gg \circ (Hbuff \circ I \parallel ch_2)^{\cup} \sqsubseteq dly \\
& \quad \wedge \;\; ch_2 \sqcap \gg \circ (ch_1 \circ Hbuff)^{\cup} \sqsubseteq dly \circ eqsp \\
= & \quad \{ \text{ reverse } \} \\
& ch_1 \sqcap (Hbuff \circ I \parallel ch_2 \circ \gg^{\cup})^{\cup} \sqsubseteq dly \\
& \quad \wedge \;\; ch_2 \sqcap (ch_1 \circ Hbuff \circ \gg^{\cup})^{\cup} \sqsubseteq dly \circ eqsp \\
= & \quad \{ \text{ reversed Parallel computation 3.32d } \} \\
& ch_1 \sqcap (Hbuff \circ \gg^{\cup} \circ ch_2)^{\cup} \sqsubseteq dly \\
& \quad \wedge \;\; ch_2 \sqcap (ch_1 \circ Hbuff \circ \gg^{\cup})^{\cup} \sqsubseteq dly \circ eqsp \\
\Leftarrow & \quad \{ \text{ Lemma 12.16 } \} \\
& ch_1 \sqcap (pref \circ click \circ ch_2)^{\cup} \sqsubseteq dly \\
& \quad \wedge \;\; ch_2 \sqcap (ch_1 \circ pref \circ click)^{\cup} \sqsubseteq dly \circ eqsp
\end{array}
$$

The hot buffer is reduced to one of its important parts: the proc *click*. We continued the calculation by applying the substitution, resulting in subexpressions $pref \circ click \circ ch_2$ and $ch_1 \circ pref \circ click$. In these expressions, some channels can be removed by exploiting Conditions 14.4a and 14.4c:

$$
\begin{array}{ll}
& pref \circ click \circ ch_2 \\
\sqsubseteq & \quad \{ \text{ Condition 14.4c } \} \\
& pref \circ click \circ pref \circ eqsp \\
\sqsubseteq & \quad \{ \text{ Axiom 12.13c } \} \\
& pref \circ pref \circ click \circ eqsp \\
= & \quad \{ \text{ Theorem 10.16f and Axiom 12.11a } \} \\
& pref \circ click
\end{array}
$$

And the other subexpression:

$$
\begin{array}{ll}
 & ch_1 \circ pref \circ click \\
\sqsubseteq & \quad \{ \text{ Condition 14.4a } \} \\
 & pref \circ eqsp \circ pref \circ click \\
= & \quad \{ \text{ Theorem 10.18 } \} \\
 & pref \circ pref \circ eqsp \circ click \\
= & \quad \{ \text{ Theorem 10.18 and Axiom 12.11a } \} \\
 & pref \circ click
\end{array}
$$

Adding up the intermediate results leads us to Conditions 14.4b and 14.4d:

$$
\begin{array}{ll}
 & ch_1 \sqcap (pref \circ click \circ ch_2)^{\cup} \sqsubseteq dly \\
 & \quad \wedge \; ch_2 \sqcap (ch_1 \circ pref \circ click)^{\cup} \sqsubseteq dly \circ eqsp \\
\Leftarrow & \quad \{ \text{ Conditions 14.4a and 14.4c: above results } \} \\
 & ch_1 \sqcap (pref \circ click)^{\cup} \sqsubseteq dly \\
 & \quad \wedge \; ch_2 \sqcap (pref \circ click)^{\cup} \sqsubseteq dly \circ eqsp \\
= & \quad \{ \text{ heading for Definition 12.17: Conditions 14.4a and 14.4c again } \} \\
 & ch_1 \sqcap eqsp \circ pref \sqcap (pref \circ click)^{\cup} \sqsubseteq dly \\
 & \quad \wedge \; ch_2 \sqcap eqsp \circ pref \sqcap (pref \circ click)^{\cup} \sqsubseteq dly \circ eqsp \\
\Leftarrow & \quad \{ \text{ below } \} \\
 & ch_1 \sqcap eqsp \circ (pref \sqcap (pref \circ click)^{\cup}) \sqsubseteq dly \\
 & \quad \wedge \; ch_2 \sqcap eqsp \circ (pref \sqcap (pref \circ click)^{\cup}) \sqsubseteq dly \circ eqsp \\
= & \quad \{ \text{ Definition 12.17a } \} \\
 & ch_1 \sqcap eqsp \circ (1) \sqsubseteq dly \\
 & \quad \wedge \; ch_2 \sqcap eqsp \circ (1) \sqsubseteq dly \circ eqsp \\
= & \quad \{ \text{ Conditions 14.4b and 14.4d } \} \\
 & True
\end{array}
$$

Only one proof obligation is left:

$$
\begin{array}{ll}
 & eqsp \circ pref \sqcap (pref \circ click)^{\cup} \sqsubseteq eqsp \circ (pref \sqcap (pref \circ click)^{\cup}) \\
\Leftarrow & \quad \{ \text{ Dedekind 3.21 } \} \\
 & pref \sqcap eqsp^{\cup} \circ (pref \circ click)^{\cup} \sqsubseteq pref \sqcap (pref \circ click)^{\cup} \\
= & \quad \{ \text{ reverse } \} \\
 & pref \sqcap (pref \circ click \circ eqsp)^{\cup} \sqsubseteq pref \sqcap (pref \circ click)^{\cup} \\
= & \quad \{ \text{ Axiom 12.11a } \} \\
 & True
\end{array}
$$

□

The importance of the use of Lemma 14.3 in the proof of Lemma 14.5 can not be emphasised enough: the key step in the proof of Lemma 14.3 is Theorem 12.10c, but due to the sequential composition with ch_1 in the Basic Network, direct application of the theorem in the proof of Lemma 14.5 (second step) is impossible.
Next, we turn to the second part needed to get the required refinement: totality of (a typed version of) $BN.(ch_1, ch_2)$. The property of totality is hard to explain as a safety or

liveness property. Had we considered the refusal of a message as observable, the totality would have been a safety property.
To be able to continue our development, we have to introduce type conditions. We will show that, under reasonable conditions, the proc $BN.(ch_1, ch_2) \circ I_a$ is total on I_a. Consequently, we can derive that this typed Basic Network is a refinement of dly_a. The intended meaning of base type a is that it is the type of the messages which are sent to the protocol.
Because the Basic Network is a feedback loop, we need causality conditions according to Corollary 13.1e. The following (weak) causality conditions are imposed on the channels ch_1 and ch_2:

Condition 14.6 *Causality of one-place channel*

a. $ch_1 \in I_a \overset{c}{\rightsquigarrow} I_a$

b. $ch_2 \circ I_a \in I_b \overset{w}{\rightsquigarrow} I_a$

□

Base type b is, for the moment, arbitrary. It is used internally by the protocol to trigger the hot buffer. When we continue our development of the protocol in Section 14.2, the intended meaning of b will become clear.
Condition 14.6 and Lemma 14.5, which is derived from Condition 14.4, guide the way to an important theorem. We want to show that the typed network $BN.(ch_1, ch_2) \circ I_a$ implements a reliable channel, using channels ch_1 and ch_2 which might loose messages:

Theorem 14.7 *Refinement and causality*

- a. $BN.(ch_1, ch_2) \circ I_a \leq dly_a$
- b. $causal.(BN.(ch_1, ch_2) \circ I_a)$

Proof:
To obtain the refinement, only totality of the Basic Network has to be proved:

$$
\begin{array}{ll}
 & BN.(ch_1, ch_2) \circ I_a \leq dly_a \\
= & \quad \{ \text{ Definitions 7.31 and 13.4c } \} \\
 & BN.(ch_1, ch_2) \circ I_a \sqsubseteq dly \circ I_a \ \wedge \ (BN.(ch_1, ch_2) \circ I_a)^{>} = dly_a{}^{>} \\
= & \quad \{ \text{ Lemma 14.5: monotonicity; Theorem 13.5d: } dly_a{}^{>} = I_a \ \} \\
 & (BN.(ch_1, ch_2) \circ I_a)^{>} = I_a
\end{array}
$$

Totality and causality of the typed Basic Network are proved together. We are heading for an application of Corollary 13.1e:

$$
\begin{array}{ll}
 & BN.(ch_1, ch_2) \circ I_a \in I_a \overset{c}{\rightsquigarrow} I_a \\
= & \quad \{ \text{ Definition 14.2 } \} \\
 & (ch_1 \circ Hbuff \circ I \parallel ch_2)^{\sigma} \circ I_a \ \in \ I_a \overset{c}{\rightsquigarrow} I_a \\
= & \quad \{ \text{ Condition 14.6a: } ch_1 = I_a \circ ch_1 \text{; Theorem 3.15 } \} \\
 & (I_a \circ ch_1 \circ Hbuff \circ I_a \parallel ch_2)^{\sigma} \in I_a \overset{c}{\rightsquigarrow} I_a
\end{array}
$$

$$
\begin{array}{ll}
= & \quad \{\ (A \circ P)^\sigma \ = \ (P \circ I \parallel A)^\sigma \ \} \\
 & (ch_1 \ \circ \ Hbuff \ \circ \ I_a \parallel (ch_2 \circ I_a))^\sigma \ \in \ I_a \overset{c}{\rightsquigarrow} I_a \\
\Leftarrow & \quad \{\ \text{Corollary 13.1e} \ \} \\
 & ch_1 \ \circ \ Hbuff \ \circ \ I_a \parallel (ch_2 \circ I_a) \ \in \ I_a \overset{c}{\rightsquigarrow} I_a \parallel I_a
\end{array}
$$

We need the causality and totality of the typed proc $ch_1 \circ Hbuff \circ I_a \parallel (ch_2 \circ I_a)$. This can be derived from Condition 14.6:

$$
\begin{array}{ll}
 & ch_1 \ \circ \ Hbuff \ \circ \ I_a \parallel (ch_2 \circ I_a) \ \in \ I_a \overset{c}{\rightsquigarrow} I_a \parallel I_a \\
\Leftarrow & \quad \{\ \text{Corollary 13.1a} \ \} \\
 & ch_1 \in I_a \overset{c}{\rightsquigarrow} I_a \ \wedge \ Hbuff \ \circ \ I_a \parallel (ch_2 \circ I_a) \ \in \ I_a \overset{w}{\rightsquigarrow} I_a \parallel I_a \\
= & \quad \{\ \text{Condition 14.6a; Condition 14.6b: } ch_2 \circ I_a = I_b \circ ch_2 \circ I_a \ \} \\
 & Hbuff \ \circ \ I_a \parallel (I_b \circ ch_2 \circ I_a) \ \in \ I_a \overset{w}{\rightsquigarrow} I_a \parallel I_a \\
\Leftarrow & \quad \{\ \text{Parallel-parallel fusion 3.12b; Corollary 13.1b} \ \} \\
 & Hbuff \circ I_a \parallel I_b \ \in \ I_a \overset{w}{\rightsquigarrow} I_a \parallel I_b \ \wedge \ I_a \parallel (ch_2 \circ I_a) \ \in \ I_a \parallel I_b \overset{w}{\rightsquigarrow} I_a \parallel I_a \\
\Leftarrow & \quad \{\ \text{Definition 13.4d and Theorem 13.5e; Corollary 13.1d} \ \} \\
 & I_a \in I_a \overset{w}{\rightsquigarrow} I_a \ \wedge \ ch_2 \circ I_a \in I_b \overset{w}{\rightsquigarrow} I_a \\
= & \quad \{\ \text{Theorem 13.5a; Condition 14.6b} \ \} \\
 & True
\end{array}
$$

□

This concludes our preliminary investigations of the Basic Network. We derived the fundamental properties of channels ch_1 and ch_2 which guarantee the well-behavedness of the network in the sense of Theorem 14.7, and removed the topological structure of the loop.

14.2 Tagged messages

In the previous section we considered a network, Figure 14.2, which was a simplification of the topology of the *ABP*, Figure 14.1. In this section, an important detail of the processes (programs) is introduced which results in the implementation of channels ch_1 and ch_2 by one single channel ch_c for some base type c.
Because the *ABP* has to be built with two faulty channels *flch* which might lose messages, we are heading for the use of a proc which repeatedly sends a message to the unreliable channel. To be able to send two equal messages to the protocol one after the other, we have to find a way to distinguish them. Otherwise, they are lost in the repetition proc. This suggests the mechanism of *tagging*. In the sequel, base type b will be used for the type of tags.
We first concentrate on the interface *nad* which is the identity on chronicles that carry *no adjacent duplicates*, regardless of the arity of the chronicle. Among others, chronicles carrying 0 and 1 in alternating order are included in the interface *nad*. Recalling that the interface *synp* is an identity on synchronised pairs, we conclude that the interface $I \parallel nad \circ synp$ is the identity on synchronised pairs such that the second components of

two adjacent pairs are non-equal. A consequence is that two adjacent synchronised pairs are distinguishable. This is formally described by the next axiom:

Axiom 14.8 *No adjacent duplicates*

$$I \parallel nad \circ synp \sqsubseteq nad \sqsubseteq I$$

□

With this knowledge, we define a proc *tag* which adds an extra message (a tag) to its argument message to assure that all outgoing adjacent messages are different:

Definition 14.9 *Acknowledge and tag*

a. $ackn \triangleq nad \circ eqsp \;\;,\;\; ackn_{ba} \triangleq I_b \circ ackn \circ I_a$

b. $tag \triangleq I \triangle ackn \;\;,\;\; tag_{ab} \triangleq I_a \parallel I_b \circ tag \;\; (= I_a \triangle ackn_{ba})$

□

The proc *ackn* replaces any input message by a new message such that no two adjacent messages are equal. The proc *tag* does not change the occurrence of messages: for every message it immediately produces a fresh tag and attaches this tag to the message. This is expressed by the inclusion $tag \sqsubseteq eqsp$:

Lemma 14.10

a. $tag \sqsubseteq eqsp$

b. $tag \sqsubseteq nad \circ synp \circ \ll^{\cup}$

Proof:

$$\begin{array}{ll} & tag \\ = & \quad \{ \text{ Definition 14.9 } \} \\ & I \triangle (nad \circ eqsp) \\ \sqsubseteq & \quad \{ \text{ } nad \text{ is an interface } \} \\ & I \triangle eqsp \\ \sqsubseteq & \quad \{ \text{ Theorem 5.18a } \} \\ & eqsp \circ I \triangle eqsp \\ = & \quad \{ \text{ Lemma 11.9 } \} \\ & eqsp \end{array}$$

The second property follows from $I \triangle eqsp = synp \circ \ll^{\cup}$ and Axiom 14.8
□

The instruments of *nad*, *ackn* and *tag* are sufficient to reason about tagging messages. Next, we show how the channels ch_1 and ch_2 can be implemented by one channel ch_c in combination with the procs *ackn* and *tag*.
For the channel ch_1 for sending the messages in the Basic Network we want to use $\ll \circ ch_{a \times b} \circ tag$: tag the messages, sent them to the channel $ch_{a \times b}$ and strip off the tags

when the tagged messages are transmitted. The channel ch_2 for the acknowledgements is less stringent. The proc $ch_b \circ ackn_{ba}$ is proposed: for every input message, generate a fresh acknowledge which is sent to the channel. To meet the Conditions 14.4 and 14.6 of Theorem 14.7, we assume the following about ch_c:

Condition 14.11 *Weak one-place channel* ch_c
For $c \in \{b, a \times b\}$:

a. $ch_c \sqsubseteq eqsp \circ pref$

b. $ch_c \circ nad \sqcap (1) \circ eqsp \sqsubseteq dly$

c. $ch_c \in I_c \overset{c}{\rightsquigarrow} I_c$

□

Conditions 14.11a and 14.11b are weaker than Conditions 14.4a and 14.4b: the channel only has to behave as a one-place channel on chronicles carrying no adjacent duplicates. Condition 14.11c is the usual healthiness condition on channel ch_c.
It follows that the number of messages produced by the proposed channels is less than the number of received input messages, or, in other words, they meet Conditions 14.4a and 14.4c:

Lemma 14.12

a. $\ll \circ ch_{a\times b} \circ tag \sqsubseteq eqsp \circ pref$

b. $ch_b \circ ackn \sqsubseteq eqsp \circ pref$

Proof:

$\ll \circ ch_{a\times b} \circ tag$
$=$ { Condition 14.11c and Theorem 13.3b: $ch_{a\times b}{}^{<} \sqsubseteq I_{a\times b} \sqsubseteq synp$ }
$\ll \circ synp \circ ch_{a\times b} \circ tag$
$\sqsubseteq$ { Condition 14.11a }
$\ll \circ synp \circ eqsp \circ pref \circ tag$
$=$ { relational calculus: $\ll \circ synp = (I \triangleleft eqsp)^{\cup}$; reverse and Lemma 11.9 }
$eqsp \circ pref \circ tag$
$\sqsubseteq$ { Lemma 14.10a }
$eqsp \circ pref \circ eqsp$
$=$ { Theorem 10.18 twice }
$eqsp \circ pref$

Lemma 14.12b follows by an easier calculation from Condition 14.11a only
□

The interface *nad* in the proc *tag* does not play any role in the proof above. It will play an important role in the next lemma, which exploits Condition 14.11b. The lemma proves

that the proposed channels meet Conditions 14.4b and 14.4d:

Lemma 14.13

a. $\ll \circ ch_{a\times b} \circ tag \sqcap (1) \circ eqsp \sqsubseteq dly$

b. $ch_b \circ ackn \sqcap (1) \circ eqsp \sqsubseteq dly \circ eqsp$

Proof:

$$
\begin{array}{ll}
 & \ll \circ ch_{a\times b} \circ tag \sqcap (1) \circ eqsp \\
\sqsubseteq & \quad \{ \text{ Lemma 14.10b } \} \\
 & \ll \circ ch_{a\times b} \circ nad \circ \ll^{\cup} \sqcap (1) \circ eqsp \\
= & \quad \{ \text{ Condition 14.11c and Theorem 13.3b: } ch_{a\times b} = synp \circ ch_{a\times b} \circ synp \ \} \\
 & \ll \circ synp \circ ch_{a\times b} \circ nad \circ synp \circ \ll^{\cup} \sqcap (1) \circ eqsp \\
= & \quad \{ \ \ll \text{ is a function: Theorem 3.35b twice } \} \\
 & \ll \ \circ \ (synp \circ ch_{a\times b} \circ nad \circ synp \sqcap \ll^{\cup} \circ (1) \circ eqsp \circ \ll) \ \circ \ \ll^{\cup} \\
= & \quad \{ \ A \circ P \circ B \sqcap Q \ = \ P \sqcap A \circ Q \circ B \ \} \\
 & \ll \ \circ \ (ch_{a\times b} \circ nad \sqcap synp \circ \ll^{\cup} \circ (1) \circ eqsp \circ \ll \circ synp) \ \circ \ \ll^{\cup} \\
\sqsubseteq & \quad \{ \text{ below } \} \\
 & \ll \ \circ \ (ch_{a\times b} \circ nad \sqcap (1) \circ eqsp) \ \circ \ \ll^{\cup} \\
\sqsubseteq & \quad \{ \text{ Condition 14.11b } \} \\
 & \ll \circ \ dly \circ \ll^{\cup} \\
= & \quad \{ \ \ll = \ll^{\circ} \text{: Axiom 12.19h } \} \\
 & dly \circ \ll \circ \ll^{\cup} \\
\sqsubseteq & \quad \{ \ \ll \text{ is a function } \} \\
 & dly
\end{array}
$$

To discharge the deferred proof obligation we use the inclusion $I \vartriangle eqsp \sqsubseteq eqsp$, which is a consequence of Lemma 11.9:

$$
\begin{array}{ll}
 & synp \circ \ll^{\cup} \circ (1) \circ eqsp \circ \ll \circ synp \\
= & \quad \{ \ synp \circ \ll^{\cup} = I \vartriangle eqsp \text{; reverse } \} \\
 & I \vartriangle eqsp \circ (1) \circ eqsp \circ (I \vartriangle eqsp)^{\cup} \\
\sqsubseteq & \quad \{ \ I \vartriangle eqsp \sqsubseteq eqsp \text{; reverse } \} \\
 & eqsp \circ (1) \circ eqsp \circ eqsp \\
= & \quad \{ \text{ Theorem 12.18e; Theorem 5.18c } \} \\
 & (1) \circ eqsp
\end{array}
$$

Lemma 14.13b follows from Definition 14.9a and an appeal to Dedekind 3.21
□

A condition is needed to guarantee totality of the acknowledge proc $ackn_{ba}$ and of the tagging proc tag_{ab}. Condition 14.14 describes that given any support, one can construct a

chronicle in I_b which carries no adjacent duplicates. Furthermore, we assume that the proc $I_b \circ ackn$ is weakly causal:

Condition 14.14 *Acknowledge*

$$I_b \circ ackn \in I \overset{w}{\rightsquigarrow} I$$

□

Observe that the condition implies that there are at least two different messages in b. Due to the type restriction I_b, weak causality of $I_b \circ ackn$ can not be derived from the weak causality of I_b and $ackn$, exploiting Axiom 9.8q.
Now the typing of the acknowledge proc $ackn_{ba}$ and of the tagging proc tag_{ab} can be derived:

Lemma 14.15

a. $ackn_{ba} \in I_b \overset{w}{\rightsquigarrow} I_a$

b. $tag_{ab} \in I_{a \times b} \overset{w}{\rightsquigarrow} I_a$

Proof:

$tag_{ab} \in I_{a \times b} \overset{w}{\rightsquigarrow} I_a$
$=$ { Theorem 13.3b }
$tag_{ab} \in I_a \parallel I_b \circ synp \overset{w}{\rightsquigarrow} I_a$
$=$ { Definition 14.9b and Lemma 14.10b: $(tag_{ab})^{<} \sqsubseteq synp$ }
$tag_{ab} \in I_a \parallel I_b \overset{w}{\rightsquigarrow} I_a$
$=$ { Definition 14.9b }
$I_a \vartriangle ackn_{ba} \in I_a \parallel I_b \overset{w}{\rightsquigarrow} I_a$
$\Leftarrow$ { Corollary 13.1c }
$I_a \in I_a \overset{w}{\rightsquigarrow} I_a \wedge ackn_{ba} \in I_b \overset{w}{\rightsquigarrow} I_a$
$=$ { Theorem 13.5a; Lemma 14.15a }
$True$

□

Finally, according to Condition 14.6, the causality of channel $\ll \circ ch_{a \times b} \circ tag$ and the weak causality of $ch_b \circ ackn \circ I_a$ has to be shown:

Lemma 14.16

a. $\ll \circ ch_{a \times b} \circ tag \in I_a \overset{c}{\rightsquigarrow} I_a$

b. $ch_b \circ ackn \circ I_a \in I_b \overset{w}{\rightsquigarrow} I_a$

Proof:

$\ll \circ ch_{a \times b} \circ tag \in I_a \overset{c}{\rightsquigarrow} I_a$
$=$ { Condition 14.11c, Theorem 13.3b: $ch_{a \times b} = I_a \parallel I_b \circ ch_{a \times b} \circ I_a \parallel I_b$ }

$$
\begin{array}{ll}
 & \ll \circ I_a \parallel I_b \circ ch_{a\times b} \circ I_a \parallel I_b \circ tag \ \in \ I_a \overset{c}{\rightsquigarrow} I_a \\
= & \quad \{ \text{ Definitions 13.4a and 14.9b } \} \\
 & \ll_{ab} \circ ch_{a\times b} \circ tag_{ab} \in I_a \overset{c}{\rightsquigarrow} I_a \\
\Leftarrow & \quad \{ \text{ Corollary 13.1a } \} \\
 & \ll_{ab} \in I_a \overset{w}{\rightsquigarrow} I_a \parallel I_b \ \wedge \ ch_{a\times b} \circ tag_{ab} \in I_a \parallel I_b \overset{c}{\rightsquigarrow} I_a \\
\Leftarrow & \quad \{ \text{ Theorem 13.5b; Corollary 13.1a } \} \\
 & ch_{a\times b} \in I_a \parallel I_b \overset{c}{\rightsquigarrow} I_{a\times b} \ \wedge \ tag_{ab} \in I_{a\times b} \overset{w}{\rightsquigarrow} I_a \\
\Leftarrow & \quad \{ \text{ Theorem 13.3b: } I_a \parallel I_b \sqsupseteq I_{a\times b} \text{; Lemma 14.15b } \} \\
 & ch_{a\times b} \in I_{a\times b} \overset{c}{\rightsquigarrow} I_{a\times b} \\
= & \quad \{ \text{ Condition 14.11c } \} \\
 & \mathit{True}
\end{array}
$$

And second:

$$
\begin{array}{ll}
 & ch_b \circ ackn \circ I_a \in I_b \overset{w}{\rightsquigarrow} I_a \\
\Leftarrow & \quad \{ \text{ Axiom 9.8n } \} \\
 & ch_b \circ ackn \circ I_a \in I_b \overset{c}{\rightsquigarrow} I_a \\
= & \quad \{ \text{ Condition 14.11c: } ch_b = ch_b \circ I_b \ \} \\
 & ch_b \circ I_b \circ ackn \circ I_a \in I_b \overset{c}{\rightsquigarrow} I_a \\
= & \quad \{ \text{ Definition 14.9a } \} \\
 & ch_b \circ ackn_{ba} \in I_b \overset{c}{\rightsquigarrow} I_a \\
\Leftarrow & \quad \{ \text{ Corollary 13.1a } \} \\
 & ch_b \in I_b \overset{c}{\rightsquigarrow} I_b \ \wedge \ ackn_{ba} \in I_b \overset{w}{\rightsquigarrow} I_a \\
= & \quad \{ \text{ Condition 14.11c; Lemma 14.15a } \} \\
 & \mathit{True}
\end{array}
$$

□

This shows that the channels are (weakly) causal, and therefore the requirements of Theorem 14.7 are met. We conclude this section with the corollary of Condition 14.11. It shows how the two channels ch_1 and ch_2 of the Basic Network can be implemented by the two channels $\ll \circ ch_{a\times b} \circ tag$ and $ch_b \circ ackn$:

Corollary 14.17 *Basic Network*

- a. $BN.(\ll \circ ch_{a\times b} \circ tag \ , \ ch_b \circ ackn) \circ I_a \leq dly_a$
- b. $causal.(BN.(\ll \circ ch_{a\times b} \circ tag \ , \ ch_b \circ ackn) \circ I_a)$

□

As remarked before, Conditions 14.4b and 14.4d (and consequently Condition 14.11b) are often too stringent. In the next section we will try to meet these conditions using a less reliable channel *flch* and preprocessing and postprocessing by additional procs to circumvent the unreliability.

14.3 The Alternating Bit Protocol

In this section, an implementation of channel ch_c is suggested, given a faulty channel flch_c. Afterwards, a sender S and a receiver R are derived which result in an instantiation of the network in Figure 14.1.

14.3.1 Weak fairness

In Condition 14.11 we required that the channel ch_c behaves like a weak one-place channel. However, when a channel can lose a message also if there is no next message, one can not meet this requirement directly. The solution is to repeatedly send the same message to the channel. A fairness property then has to prescribe that eventually at least one message is transmitted by the channel. A proc placed behind the channel removes the duplicates generated by the repetition proc.
Given a faulty channel flch_c, we want to specify a repetition proc rep_c and a removing proc rem_c such that the sequential composition $\mathit{rem}_c \circ \mathit{flch}_c \circ \mathit{rep}_c$ satisfies Condition 14.11 needed to establish the result of Corollary 14.17. To meet those conditions, we assume that $\mathit{rem}_c \circ \mathit{flch}_c \circ \mathit{rep}_c$ is a weak one-place channel:

Condition 14.18 *Weak one-place channel*
For $c \in \{b, a \times b\}$:

$\mathit{rem}_c \circ \mathit{flch}_c \circ \mathit{rep}_c$ is a weak one-place channel

□

We are obliged to shed some light on Condition 14.18 and in particular on Condition 14.11b with ch_c replaced by $\mathit{rem}_c \circ \mathit{flch}_c \circ \mathit{rep}_c$. The explanation is best understood when one considers a single input message m which is sent to the channel $\mathit{rem}_c \circ \mathit{flch}_c \circ \mathit{rep}_c$. Because there is only one input m, the conjunction with the one-place buffer $(1) \circ \mathit{eqsp}$ and the inclusion in dly requires that the message m has to be transmitted eventually. Therefore, in the worst case, the part $\mathit{rem}_c \circ \mathit{flch}_c \circ \mathit{rep}_c$ has to take care for the transmission: *if the message m is sent repeatedly, the channel flch_c eventually has to transmit that message at least once.* The proc rep_c is responsible for the repetition; rem_c removes duplicates that are transmitted. In scheduling processes, this liveness property is called *weak fairness.*
The causality of the channel $\mathit{rem}_c \circ \mathit{flch}_c \circ \mathit{rep}_c$, Condition 14.11c, is implied by the conditions that the procs rem_c and rep_c are weakly causal, and the condition that flch_c is causal.
Observe that the faulty channel flch_c is allowed to produce duplicates: the proc rem_c removes those duplicates. This motivates why a condition like $\mathit{flch}_c \sqsubseteq \mathit{eqsp} \circ \mathit{pref}$, Condition 14.11a, is too strong.
Several conditions on the procs rep_c, flch_c and rem_c are combined. We do not require explicitly that the proc rep_c keeps on repeating its input. In combination with a channel flch_c which really loses messages, repetition is necessary. In case the channel flch_c is FIFO, for example if $\mathit{flch}_c = \mathit{dly}_c$, the proc pdly_c is a good instantiation for rep_c and rem_c: the obtained combination of procs satisfies Condition 14.18 trivially.

The direct corollary of Condition 14.18 and Corollary 14.17 is stated next:

Corollary 14.19 *Basic Network*

- a. $BN.(\ll \circ rem_{a\times b} \circ flch_{a\times b} \circ rep_{a\times b} \circ tag\ ,\ rem_b \circ flch_b \circ rep_b \circ ackn) \circ I_a \leq dly_a$
- b. $causal.(BN.(\ll \circ rem_{a\times b} \circ flch_{a\times b} \circ rep_{a\times b} \circ tag\ ,\ rem_b \circ flch_b \circ rep_b \circ ackn) \circ I_a)$

□

Next, we massage this instantiation of channel ch_c to obtain a familiar protocol.

14.3.2 Sender and receiver defined

Finally, we aim for our goal of finding a sender and receiver such that the topological structure of Figure 14.1 implements a delay. From Corollary 14.19 we conclude that the Basic Network with the instantiation $\ll \circ rem \circ flch \circ rep \circ tag$ and $rem \circ flch \circ rep \circ ackn$ refines a delay. To obtain a sender and a receiver, the Basic Network of Corollary 14.19 is transformed such that we get the topology of the *ABP*, Figure 14.1:

$$
\begin{array}{ll}
 & BN.(\ll \circ rem_{a\times b} \circ flch_{a\times b} \circ rep_{a\times b} \circ tag\ ,\ rem_b \circ flch_b \circ rep_b \circ ackn) \circ I_a \\
= & \quad \{\ \text{Definition 14.2; Theorem 3.15}\ \} \\
 & (\ll \circ\ rem_{a\times b} \circ\ flch_{a\times b} \circ\ rep_{a\times b} \circ\ tag \circ\ Hbuff\ \circ I_a \parallel (rem_b \circ flch_b \circ rep_b \circ ackn))^{\sigma} \\
= & \quad \{\ \text{Theorem 3.18a}\ \} \\
 & (I \triangle I \circ \ll \circ\ rem_{a\times b} \circ flch_{a\times b} \circ rep_{a\times b} \circ tag \circ Hbuff \circ I_a \parallel (rem_b \circ flch_b \circ rep_b \circ ackn))^{\varpi} \\
= & \quad \{\ \text{Parallel-parallel fusion 3.12b}\ \} \\
 & (I \triangle I \circ \ll \circ\ rem_{a\times b} \circ flch_{a\times b} \circ rep_{a\times b} \circ tag \circ Hbuff \circ I_a \| rem_b \circ I \| flch_b \circ I \parallel (rep_b \circ ackn))^{\varpi} \\
= & \quad \{\ \text{Theorem 3.18c}\ \} \\
 & (I \parallel (rep_b \circ ackn) \circ I \triangle I \circ \ll \circ\ rem_{a\times b} \circ flch_{a\times b} \circ rep_{a\times b} \circ tag \circ Hbuff \circ I_a \| rem_b \circ I \| flch_b)^{\varpi} \\
= & \quad \{\ \text{Parallel-split fusion 3.12a}\ \} \\
 & (I \triangle (rep_b \circ ackn) \circ \ll \circ\ rem_{a\times b} \circ flch_{a\times b} \circ rep_{a\times b} \circ tag \circ Hbuff \circ I_a \| rem_b \circ I \| flch_b)^{\varpi} \\
= & \quad \{\ \text{Definition 14.1}\ \} \\
 & ABP_{a\times b\, b}.(I \triangle (rep_b \circ ackn)\ \circ\ \ll\ \circ\ rem_{a\times b}\ ,\ rep_{a\times b} \circ\ tag \circ\ Hbuff\ \circ I_a \parallel rem_b)
\end{array}
$$

The derived definitions for a sender and a receiver such that $ABP.(R, S)$ implements a delay are:

Definition 14.20 *Sender S and receiver R*

a. $S \triangleq rep_{a\times b} \circ tag \circ Hbuff \circ I_a \parallel rem_b$

b. $R \triangleq I \triangle (rep_b \circ ackn)\ \circ\ \ll\ \circ\ rem_{a\times b}$

□

Initially, the sender S is in a state with one request pending, due to the hot buffer *Hbuff*. The sender keeps all incoming messages, which are received on its first input port, in the buffer and repeatedly sends those messages, after adding a fresh tag, to the output port. Whenever S gets a trigger on its second input port, the hot buffer gives a new message

to be sent if it is present; otherwise, the sender goes again to a state with one pending request.
The receiver R removes all duplicates from its input port and strips off the tags. The resulting messages are sent to its first output port and an acknowledgement is repeatedly sent to the second output port to signal that a message has been received.
The Alternating Bit Protocol theorem is stated next. It follows from Condition 14.18, Corollary 14.19 and the derivation of R and S above:

Theorem 14.21 *The ABP Theorem*

- a. $ABP_{a\times b\, b}.(R,S) \leq dly_a$
- b. $causal.ABP_{a\times b\, b}.(R,S)$

□

We finish the theoretical discussion on the ABP with a remark about the angelic sequential composition used in the protocol. All the angelic compositions can be replaced by demonic compositions if we assume the following totality condition:

Condition 14.22 *Typing*
For $c \in \{b, a\times b\}$:

$$rem_c, flch_c, rep_c \in I_c \rightsquigarrow I_c$$

□

The result then follows from type pushing and Corollary 3.45a:

Theorem 14.23 *Demonic compositions*

a. $$S \;=\; rep_{a\times b} \mathbin{\natural} tag_{ab} \mathbin{\natural} Hbuff_{ab} \mathbin{\natural} I_a \parallel rem_b$$

b. $$R \;=\; I_a \mathbin{\triangle} (rep_b \mathbin{\natural} ackn_{ba}) \mathbin{\natural} \ll_{ab} \mathbin{\natural} rem_{a\times b}$$

c. $$ABP_{a\times b\, b}.(R,S) \;=\; (R \mathbin{\natural} flch_{a\times b} \mathbin{\natural} S \mathbin{\natural} I_a \parallel flch_b)^{\varpi}$$

□

After having identified the network of Figure 14.2 as fundamental for protocols which rely on the mechanism of acknowledgements, we derived reasonable conditions on the channels of this network to guarantee the implementation of a reliable channel. It was shown that, given a faulty channel, those conditions can be met by a suitable preprocessing and postprocessing of the messages to be sent. This resulted in a possible instantiation of Figure 14.1 which implements a delay.
Observe that the name 'Alternating Bit Protocol' is too specific: it is not essential that alternating bits be used as tags. Any sequence of pairwise distinguishable tags will do; this is formalised by the use of interface *nad* and Axiom 14.8.

Chapter 15

Conclusion and future work

We draw some conclusions from the preceding investigations, and present some suggestions for future work, either as an extension of the theory presented in this thesis, or as an alternative.

15.1 Conclusion

Our goal was, and is, the development of a relational calculus for the design of distributed algorithms. After intensive research, of which this thesis is the result, the question should be asked: How far did we get?
The choice for starting with a well-established relational calculus was a significant leap forward. Due to work done by other researchers, Part I records an extensive set of theorems which are at our disposal. The only problem was to pinpoint the specific model needed to reason about processes. In essence, the model is the stream model for history relations, extended with time information to be able to describe causality properties. We showed, after some preliminary work on postcompose and precompose in Part II, that the property of causality solves the preservation problems of feedback with respect to (poly-)functionality and totality. This work, presented in Part III, is the main body of the theory developed in this thesis. In Part IV, we succeeded in giving definitions (partly in the model with precompose) of several useful procs, and fixed a number of properties as axioms which can be applied during the derivation of distributed algorithms. As an example of such a derivation, Part V presents the proof of the Alternating Bit Protocol. The proof is conducted completely on the level of procs. No elements of the model occur in the calculation; only some informal justifications and clarifications are given in terms of chronicles and messages.
The tool box of the calculus contains only about ten primitive procs. The set of combining forms is small too; it mainly comprises the primitive combining forms of the raw relational calculus, extended with a few derived forms such as feedback and postcompose. Most of the axioms recorded at the end of each part fit in classes such as commutation and preservation properties.

The absence of state results in proofs which only mention processes; no internal structure is visible. Together with the goal to perform calculations point-free, we could completely concentrate on the processes as first-class citizens.
We succeeded in abstracting from time in a pleasant way: the procs which are related to assumptions on the time domain, such as *dly*, are axiomatised without explicit reference to time. As announced, different assumptions lead to different rules of the calculus.

We can, however, indicate several drawbacks of the approach. Despite the fact that the relational calculus is well developed, it does not mean that it is well known. The reason is that most research has been done using models built on traces or transition systems; a relational calculus as the basis is relatively new in the field. One calculus which applies a relational algebra is the language called Ruby, developed by Jones & Sheeran [JS90]. For us, the application of the calculus of relations for the design of distributed algorithms was a first try. Therefore, it might be the case that we made some wrong decisions, and that other choices would have led to a nicer set of theorems.
For example, the choice for an untyped calculus forced us to introduce assumptions on relations and procs such as primed typing and preservation of arities. We showed that instantiating the general theory is possible, Chapter 13, but starting off with a well-typed calculus of relations in the first place simply avoids considerations such as primed typing. On the other hand, a typed calculus introduces typing problems with cap.
The dependence of Axiom 12.19p (and all its corollaries) on Assumption 10.8 is a mistake. This is due to the restrictive definitions of *pdly* and *dly* when interpreted for $\mathcal{T}$ isomorphic to $\mathbf{Z}$.
Several calculations are very unwieldy, or simply cannot be called calculations in the first place. The 'proofs' of the causality and weak causality of the procs *eqar*, *eqsp* and *Sync* (Properties 8.1b, 9.4d and 11.11) are a thorn in the flesh of the calculus. The calculations concerning buffered procs, Theorem 12.7, were tedious and long as well.

15.2 Future work

Alternative definitions for *pdly* and *dly* establishing an order-isomorphism between the *supports* of chronicles have to be given. This will result in an intuitively correct interpretation of these procs and remove the dependence of Axiom 12.19p on Assumption 10.8. Commutation of *pdly* or *dly* with r° (Axiom 12.19g and 12.19h) will be false in general. However, under conditions like $sm \bullet r = r \bullet sm$ or Assumption 10.8 these axioms will be valid.
In Chapters 13 and 14 we introduced typed procs. These chapters suggest a subset of procs which could be valuable with respect to implementations. The rules listed in Corollary 13.1 resemble (typing) rules for a category of total and causal procs. If we are able to identify the structure of a category, we get theorems known from category theory 'for free', see Goldblatt [Gol86], MacLane [Lan88] or, more specifically, Abramsky [Abr94]. Future research has to study the effect of the type-checking and other restrictions emerging during calculations.

Another important possibility suggested by category theory is the use of *functors* (or, in our case, relators) to describe the topology of large networks. In our calculus, the parallel composition is a functor, and therefore we claim that functors can be useful in defining and manipulating the topological structure of networks. Preliminary investigations showed that work by Jones & Sheeran [JS90] on Ruby fits in our framework, and that several topological constructs such as columns col and rows row can be defined generically by making use of initial algebras of appropriate functors.
The usefulness of the calculus could be increased by adding so-called *accumulations* which simulate an internal state: present output depends on present input *and* on the previous output. The result is a generalisation of the function postcompose which applies its argument only to the present input. For example, a proc that adds all the natural numbers on the input and delivers the intermediate results can be defined using accumulations. Early investigations on accumulations showed that there are similarities with relational catamorphisms on non-empty lists, Backhouse *et al.* [BdBM$^+$91]. Accumulations are powerful instruments, and we would like to have them in our tool box.
Further research could also investigate and encapsulate the connections with dataflow networks (see Appendix C), process algebras, delay-insensitive algebras and real-time calculi.

The final conclusion is that the first steps towards a relational calculus for the design of distributed algorithms are promising, but leave enough questions unanswered, or solved in such an unsatisfactory way that alternatives have to be found.

Bibliography

[Aar92] C.J. Aarts. Galois connections presented calculationally. Master's thesis, Eindhoven University of Technology, 1992.

[Abr90] S. Abramsky. A generalised Kahn principle for abstract asynchronous networks. In M. Main, A. Melton, and D. Schmidt, editors, *Mathematical Foundations of Program Semantics*, volume 442 of *Lecture Notes in Computer Science*, pages 1–21. Springer-Verlag, 1990.

[Abr94] S. Abramsky. Interaction categories and communicating sequential processes. In A.W. Roscoe, editor, *A Classical Mind: Essays in Honour of C.A.R. Hoare*. Prentice-Hall, 1994.

[Apo67] T.M. Apostol. *Calculus*, volume 1, second edition. Wiley International Edition, 1967.

[BA81] J.D. Brock and W.B. Ackerman. Scenarios: A model of non-deterministic computation. In J. Diaz and I. Ramos, editors, *Formalization of Programming Concepts*, volume 107 of *Lecture Notes in Computer Science*, pages 252–259. Springer-Verlag, 1981.

[Bac78] J. Backus. Can programming be liberated from the von Neumann style? a functional style and its algebra of programs. *Communications of the ACM*, 21(8):613–641, 1978.

[Bac93] R.-J.R. Back. Refinement calculus, lattices and higher order logic. In M. Broy, editor, *Program Design Calculi*, volume 118 of *Springer NATO ASI Series*, pages 53–71, 1993.

[BdBH+91] R.C. Backhouse, P.J. de Bruin, P.F. Hoogendijk, G. Malcolm, T.S. Voermans, and J.C.S.P. van der Woude. Polynomial relators. In M. Nivat, C. Rattray, T. Rus, and G. Scollo, editors, *Algebraic Methodology and Software Technology*, Workshops in Computing, pages 303–326. Springer-Verlag, 1991.

[BdBM+91] R.C. Backhouse, P.J. de Bruin, G. Malcolm, T.S. Voermans, and J.C.S.P. van der Woude. Relational catamorphisms. In B. Möller, editor, *Proceedings of the IFIP Working Conference on Constructing Programs*, pages 287–318. Elsevier Science Publishers, 1991.

[Ber91] R. Berghammer. Relational specification of data types and programs. Technical Report 9109, Universität der Bundeswehr München, Fakultät für Informatik, 1991.

[Bro90] M. Broy. Functional specification of time sensitive communicating systems. In J.W. de Bakker, W.-P. de Roever, and G. Rozenberg, editors, *Stepwise Refinement of Distributed Systems: Models, Formalisms, Correctness*, volume 430 of *Lecture Notes in Computer Science*, pages 153–179. Springer-Verlag, 1990.

[BSW69] K.A. Bartlett, R.A. Scantleburry, and P.T. Wilkinson. A note on reliable full-duplex transmission over half-duplex links. *Communications of the ACM*, 12(5):260–261, 1969.

[BvdW93] R.C. Backhouse and J.C.S.P. van der Woude. Demonic operators and monotype factors. *Mathematical Structures in Computer Science*, 3(4):417–434, 1993.

[BW88] R.S. Bird and P. Wadler. *Introduction to Functional Programming*. Prentice-Hall, 1988.

[BW90] J.C.M. Baeten and W.P. Weijland. *Process Algebra*, volume 18 of *Cambridge Tracts in TCS*. Cambridge University Press, 1990.

[Can97] G. Cantor. Beiträge zur Begründung der Transfiniten Mengenlehre. *Mathematische Annalen*, 49:207–246, 1897.

[CM89] K.M. Chandy and J. Misra. *Parallel Program Design*. Addison-Wesley, 1989.

[Doo94] H. Doornbos. A relational model of programs without the restriction to Egli-Milner monotonic constructs. In E.-R. Olderog, editor, *Programming Concepts, Methods and Calculi*, pages 363–382. North-Holland, 1994.

[DP90] B.A. Davey and H.A. Priestley. *Introduction to Lattices and Order*. Cambridge University Press, 1990.

[DS90] E.W. Dijkstra and C.S. Scholten. *Predicate Calculus and Program Semantics*. Springer-Verlag, 1990.

[Fok92] M.M. Fokkinga. *Law and Order in Algorithmics*. PhD thesis, Twente University, 1992.

[Fra86] N. Francez. *Fairness*. Springer-Verlag, 1986.

[Gol86] R. Goldblatt. *Topoi: the Categorial Analysis of Logic*. North-Holland, 1986.

[Hoa85] C.A.R. Hoare. *Communicating Sequential Processes*. Prentice-Hall, 1985.

[Jeu93] J.T. Jeuring. *Theories for Algorithm Calculation*. PhD thesis, Utrecht University, 1993.

[JS90] G. Jones and M. Sheeran. Circuit design in Ruby. In J. Staunstrup, editor, *Formal Methods for VLSI Design: IFIP WG 10.5 Lecture Notes*. North-Holland, 1990.

[Kah74] G. Kahn. The semantics of a simple language for parallel programming. In J.L. Rosenfeld, editor, *Information Processing 74: Proceedings of IFIP Congress 74*, pages 471–475. North-Holland, 1974.

[Kel78] R.M. Keller. Denotational models for parallel programs with indeterminate operators. In E.J. Neuhold, editor, *Formal Description of Programming Concepts*, pages 337–366. North-Holland, 1978.

[Kle52] S.C. Kleene. *Introduction to metamathematics*. North-Holland, 1952.

[KM68] K. Kuratowski and A. Mostowski. *Set Theory*. North-Holland, 1968.

[Kna28] B. Knaster. Un théorème sur les fonctions d'essembles. *Annales de la Societé Polonaise de Mathematique*, 6:133–134, 1928.

[Kok93] J.N. Kok. Current trends in the semantics of data flow. In M. Droste and Y. Gurevich, editors, *Semantics of Programming Languages and Model Theory*, volume 5 of *Algebra, Logic and Applications*, pages 245–268. Gordon and Breach Science Publishers, 1993.

[Lam77] L. Lamport. Proving the correctness of multiprocess programs. *IEEE Transactions on Software Engineering*, SE-3(2):125–143, 1977.

[Lan88] S. Mac Lane. *Categories for the Working Mathematician*. Springer-Verlag, 1988.

[Mad91] R.D. Maddux. The origin of relation algebras in the development and axiomatization of the calculus of relations. *Studia Logica*, 50:421–455, 1991.

[Mei92] H.J.M. Meijer. *Calculating Compilers*. PhD thesis, University of Nijmegen, 1992.

[Mil89] R. Milner. *Communication and Concurrency*. Prentice-Hall, 1989.

[Mis90] J. Misra. Equational reasoning about nondeterministic processes. *Formal Aspects of Computing*, 2:167–195, 1990.

[Möl94] B. Möller. Ideal streams. In E.-R. Olderog, editor, *Programming Concepts, Methods and Calculi*, pages 39–58. North-Holland, 1994.

[Mor90] C.C. Morgan. *Programming from Specifications*. Prentice-Hall, 1990.

[Rie91] F.J. Rietman. A note on extensionality. In J. van Leeuwen, editor, *Computing Science in the Netherlands*, pages 468–483, 1991.

[Rie93a] F.J. Rietman. The secrets of causality. Technical Report RUU-CS-93-29, Department of Computer Science, Utrecht University, 1993.

[Rie93b] F.J. Rietman. Towards the derivation of distributed algorithms in the relational calculus. In H.A. Wijshoff, editor, *Computing Science in the Netherlands*, pages 297–308, 1993.

[Rig48] J. Riguet. Relations binaires, fermetures, correspondances de galois. *Bulletin de la Societé Mathematique de France*, 76:114–155, 1948.

[Rog67] H.R. Rogers. *Theory of Recursive Functions and Effective Computability.* McGraw-Hill, New York, 1967.

[Sch53] J. Schmidt. Beiträge zur Filtertheorie. *Mathematische Nachrichten*, 10:197–232, 1953.

[Sek93] E. Sekerinski. A calculus for predicative programming. In R.S. Bird, C.C. Morgan, and J.C.P. Woodcock, editors, *Proceedings of the 2nd International Conference on the Mathematics of Program Construction*, volume 669 of *Lecture Notes in Computer Science*, pages 302–322. Springer-Verlag, 1993.

[SS93] G. Schmidt and T. Ströhlein. *Relations and graphs: discrete mathematics for computer scientists.* EATC: monographs on theoretical computer science. Springer-Verlag, 1993.

[Sta90] E.W. Stark. A simple generalization of Kahn's principle to indeterminate dataflow networks. In M.Z. Kwiatkowska, M.W. Shields, and R.M. Thomas, editors, *Semantics for Concurrency*, Workshops in Computing, pages 157–174. Springer-Verlag, 1990.

[Sta92] E.W. Stark. A calculus of dataflow networks. In *Proceedings seventh annual IEEE symposium on Logic in Computer Science*, Logic in Computer Science, pages 125–136, 1992.

[Tan81] A.S. Tanenbaum. *Computer Networks.* Prentice-Hall, 1981.

[Tar41] A. Tarski. On the calculus of relations. *Journal of Symbolic Logic*, 6(3):73–89, 1941.

[Tar55] A. Tarski. A lattice-theoretical fixpoint theorem and its applications. *Pacific Journal of Mathematics*, 5:285–309, 1955.

[Vaa90] F.W. Vaandrager. Two simple protocols. In J.C.M. Baeten, editor, *Applications of Process Algebra*, volume 17 of *Cambridge Tracts in TCS*, pages 23–44. Cambridge University Press, 1990.

[Zim80] H. Zimmermann. OSI reference model — the ISO model of architecture for open systems interconnection. *IEEE Transactions on Communications*, 28(4):425–432, 1980.

Index

Appendix A

Archimedean functions

In the discussion that follows, it is assumed that the relations α and β are *arch*, Characterisation 7.3.

Characterisation A.1 *Minimum on moments*

$$\downarrow \triangleq (\mathcal{I} \blacktriangle \geqslant \cup \geqslant \blacktriangle \mathcal{I})^{-1}$$

□

A.1 Inflating

Proposition A.2

$$\downarrow \bullet r \blacktriangle s \;=\; (r \cap \leqslant \bullet s) \cup (\leqslant \bullet r \cap s)$$

Proof:

$$\begin{array}{ll} & \downarrow \bullet r \blacktriangle s \\ = & \quad \{ \text{ Characterisation A.1 } \} \\ & (\mathcal{I} \blacktriangle \geqslant \cup \geqslant \blacktriangle \mathcal{I})^{-1} \bullet r \blacktriangle s \\ = & \quad \{ \text{ cupjunctivity of reverse and composition } \} \\ & (\mathcal{I} \blacktriangle \geqslant)^{-1} \bullet r \blacktriangle s \cup (\geqslant \blacktriangle \mathcal{I})^{-1} \bullet r \blacktriangle s \\ = & \quad \{ \text{ axiom split } \} \\ & (r \cap \geqslant^{-1} \bullet s) \cup (\geqslant^{-1} \bullet r \cap s) \\ = & \quad \{ \geqslant^{-1} = \leqslant \} \\ & (r \cap \leqslant \bullet s) \cup (\leqslant \bullet r \cap s) \end{array}$$

□

Proposition A.3

$$\downarrow \bullet \alpha \blacktriangle \beta \subseteq >$$

Proof:

$$
\begin{array}{ll}
 & \downarrow \bullet \alpha \blacktriangle \beta \\
= & \quad \{ \text{ Proposition A.2 } \} \\
 & (\alpha \cap \leqslant \bullet \beta) \cup (\leqslant \bullet \alpha \cap \beta) \\
\subseteq & \quad \{ \text{ plat calculus } \} \\
 & \alpha \cup \beta \\
\subseteq & \quad \{ \ \alpha \text{ and } \beta \text{ are inflating } \} \\
 & > \cup > \\
= & \quad \{ \text{ idempotency cup } \} \\
 & >
\end{array}
$$

□

A.2 Isomorphism

Proposition A.4

a. $(\alpha \cap \leqslant \bullet \beta) \bullet (\alpha \cap \leqslant \bullet \beta)^{-1} \;=\; \mathcal{I} \cap \alpha \bullet \geqslant \bullet \beta^{-1}$

b. $(\alpha \cap \leqslant \bullet \beta) \bullet (\leqslant \bullet \alpha \cap \beta)^{-1} \;=\; \mathcal{I} \cap \alpha \bullet \beta^{-1}$

c. $(\alpha \cap \leqslant \bullet \beta)^{-1} \bullet (\alpha \cap \leqslant \bullet \beta) \;=\; \mathcal{I} \cap \alpha^{-1} \bullet \leqslant \bullet \beta$

d. $(\alpha \cap \leqslant \bullet \beta)^{-1} \bullet (\leqslant \bullet \alpha \cap \beta) \;=\; \mathcal{I} \cap \alpha^{-1} \bullet \beta$

Proof of A.4a:

$$
\begin{array}{ll}
 & (\alpha \cap \leqslant \bullet \beta) \bullet (\alpha \cap \leqslant \bullet \beta)^{-1} \\
= & \quad \{ \ \alpha \text{ is a function } \Rightarrow (\alpha \cap \leqslant \bullet \beta) \text{ is a function } \} \\
 & (\alpha \cap \leqslant \bullet \beta)^{\prec} \\
= & \quad \{ \ (P \sqcap Q)^{\prec} = I \sqcap P \circ Q^{\cup} \ \} \\
 & \mathcal{I} \cap \alpha \bullet \beta^{-1} \bullet \geqslant \\
= & \quad \{ \text{ monotonicity of } \beta^{-1} \ \} \\
 & \mathcal{I} \cap \alpha \bullet \geqslant \bullet \beta^{-1}
\end{array}
$$

Monotonicity in the proof of A.4a is only used to get a symmetric form. The proofs of A.4b and A.4c do not require monotonicity properties. For the proof of A.4d, the monotonicity of both α and β is *needed*:

$$
\begin{array}{ll}
 & (\alpha \cap \leqslant \bullet \beta)^{-1} \bullet (\leqslant \bullet \alpha \cap \beta) \\
= & \quad \{ \text{ reverse } \} \\
 & (\alpha^{-1} \cap \beta^{-1} \bullet \geqslant) \bullet (\leqslant \bullet \alpha \cap \beta) \\
= & \quad \{ \ \alpha^{-1} \text{ is a function; } \beta \text{ is an injection } \} \\
 & \alpha^{-1} \bullet (\mathcal{I} \cap \alpha \bullet \beta^{-1} \bullet \geqslant) \bullet (\leqslant \bullet \alpha \bullet \beta^{-1} \cap \mathcal{I}) \bullet \beta \\
= & \quad \{ \text{ identities: } a \bullet b = a \cap b \ \} \\
 & \alpha^{-1} \bullet (\alpha \bullet \beta^{-1} \bullet \geqslant \cap \leqslant \bullet \alpha \bullet \beta^{-1} \cap \mathcal{I}) \bullet \beta
\end{array}
$$

$$
\begin{array}{ll}
= & \{\ \beta^{-1} \text{ and } \alpha \text{ are monotonic}\ \} \\
& \alpha^{-1} \bullet (\alpha \bullet \geqslant \bullet \beta^{-1} \cap \alpha \bullet \leqslant \bullet \beta^{-1} \cap \mathcal{I}) \bullet \beta \\
= & \{\ \alpha \text{ is an injection; } \beta^{-1} \text{ is a function}\ \} \\
& \alpha^{-1} \bullet (\alpha \bullet (\geqslant \cap \leqslant) \bullet \beta^{-1} \cap \mathcal{I}) \bullet \beta \\
= & \{\ \geqslant \cap \leqslant = \mathcal{T}\text{; types}\ \} \\
& \alpha^{-1} \bullet (\alpha \bullet \beta^{-1} \cap \mathcal{I}) \bullet \beta \\
= & \{\ \alpha^{-1} \text{ is a function; } \beta \text{ is an injection}\ \} \\
& \mathcal{I} \cap \alpha^{-1} \bullet \beta
\end{array}
$$

□

Proposition A.5

a. $\quad \downarrow \bullet \alpha \blacktriangle \beta \bullet (\downarrow \bullet \alpha \blacktriangle \beta)^{-1} = \mathcal{T}$

b. $\quad (\downarrow \bullet \alpha \blacktriangle \beta)^{-1} \bullet \downarrow \bullet \alpha \blacktriangle \beta = \mathcal{T}$

Proof of A.5a:

$$
\begin{array}{ll}
& \downarrow \bullet \alpha \blacktriangle \beta \bullet (\downarrow \bullet \alpha \blacktriangle \beta)^{-1} \\
= & \{\ \text{Proposition A.2}\ \} \\
& ((\alpha \cap \leqslant \bullet \beta) \cup (\leqslant \bullet \alpha \cap \beta)) \bullet ((\alpha \cap \leqslant \bullet \beta) \cup (\leqslant \bullet \alpha \cap \beta))^{-1} \\
= & \{\ \text{reverse}\ \} \\
& ((\alpha \cap \leqslant \bullet \beta) \cup (\leqslant \bullet \alpha \cap \beta)) \bullet ((\alpha \cap \leqslant \bullet \beta)^{-1} \cup (\leqslant \bullet \alpha \cap \beta)^{-1}) \\
= & \{\ \text{cupjunctivity}\ \} \\
& (\alpha \cap \leqslant \bullet \beta) \bullet (\alpha \cap \leqslant \bullet \beta)^{-1} \cup (\alpha \cap \leqslant \bullet \beta) \bullet (\leqslant \bullet \alpha \cap \beta)^{-1} \\
& \cup (\leqslant \bullet \alpha \cap \beta) \bullet (\alpha \cap \leqslant \bullet \beta)^{-1} \cup (\leqslant \bullet \alpha \cap \beta) \bullet (\leqslant \bullet \alpha \cap \beta)^{-1} \\
= & \{\ \text{Propositions A.4a and A.4b}\ \} \\
& (\mathcal{I} \cap \alpha \bullet \geqslant \bullet \beta^{-1}) \cup (\mathcal{I} \cap \alpha \bullet \beta^{-1}) \cup (\mathcal{I} \cap \beta \bullet \alpha^{-1}) \cup (\mathcal{I} \cap \beta \bullet \geqslant \bullet \alpha^{-1}) \\
= & \{\ \geqslant \supseteq \mathcal{T}\text{: monotonicity of cup}\ \} \\
& (\mathcal{I} \cap \alpha \bullet \geqslant \bullet \beta^{-1}) \cup (\mathcal{I} \cap \beta \bullet \geqslant \bullet \alpha^{-1}) \\
= & \{\ \mathcal{I} \cap r = \mathcal{I} \cap r^{-1}\ \} \\
& (\mathcal{I} \cap \alpha \bullet \geqslant \bullet \beta^{-1}) \cup (\mathcal{I} \cap \alpha \bullet \leqslant \bullet \beta^{-1}) \\
= & \{\ \text{cupjunctivity}\ \} \\
& \mathcal{I} \cap \alpha \bullet (\geqslant \cup \leqslant) \bullet \beta^{-1} \\
= & \{\ \geqslant \cup \leqslant = \mathcal{T} \bullet \top \bullet \mathcal{T}\text{; types}\ \} \\
& \mathcal{I} \cap \alpha \bullet \top \bullet \beta^{-1} \\
= & \{\ \alpha \text{ and } \beta \text{ are surjective to } \mathcal{T}\ \} \\
& \mathcal{T}
\end{array}
$$

The proof of A.5b follows the same structure. Propositions A.4c and A.4d, and the totality of α and β on $\mathcal{T}$ are used instead
□

A.3 Monotonic

Proposition A.6

$$\downarrow \bullet < \times < \;=\; < \bullet \downarrow$$

Proof:

$$\begin{array}{ll}
 & \downarrow \bullet < \times < \;=\; < \bullet \downarrow \\
= & \qquad \{ \text{ definition product } \} \\
 & \downarrow \bullet (< \bullet \ll) \vartriangle (< \bullet \gg) \;=\; < \bullet \downarrow \\
= & \qquad \{ \text{ Proposition A.2 } \} \\
 & (< \bullet \ll \cap \leqslant \bullet < \bullet \gg) \cup (\leqslant \bullet < \bullet \ll \cap < \bullet \gg) \;=\; < \bullet \downarrow \\
= & \qquad \{ \leqslant \bullet < \;=\; < \} \\
 & (< \bullet \ll \cap < \bullet \gg) \cup (< \bullet \ll \cap < \bullet \gg) \;=\; < \bullet \downarrow \\
= & \qquad \{ \text{ idempotency cup } \} \\
 & < \bullet \ll \cap < \bullet \gg \;=\; < \bullet \downarrow \\
= & \qquad \{ \text{ definition split; Characterisation A.1 } \} \\
 & (> \vartriangle >)^{-1} \;=\; < \bullet (\mathcal{I} \vartriangle \geqslant \cup \geqslant \vartriangle \mathcal{I})^{-1} \\
= & \qquad \{ \text{ reverse } \} \\
 & > \vartriangle > \;=\; (\mathcal{I} \vartriangle \geqslant \cup \geqslant \vartriangle \mathcal{I}) \bullet >
\end{array}$$

We are heading for case analysis. The axiom $\leqslant \cup \geqslant \;=\; \mathcal{T} \bullet \top \bullet \mathcal{T}$ is equivalent to the expression $(\mathcal{I} \vartriangle \geqslant)_{\prec} \cup (\geqslant \vartriangle \mathcal{I})_{\prec} \;=\; \mathcal{T} \times \mathcal{T}$. This equality formalises the property that for all pairs (t, t') it is either the case that $t \leqslant t'$ or $t \geqslant t'$; this is the case analysis. The expression $(\mathcal{I} \vartriangle \geqslant)_{\prec} \cup (\geqslant \vartriangle \mathcal{I})_{\prec}$ is placed after $> \vartriangle >$ in the previous calculation:

$$\begin{array}{ll}
 & > \vartriangle > \;=\; (\mathcal{I} \vartriangle \geqslant \cup \geqslant \vartriangle \mathcal{I}) \bullet > \\
= & \qquad \{ \text{ above discussion } \} \\
 & ((\mathcal{I} \vartriangle \geqslant)_{\prec} \cup (\geqslant \vartriangle \mathcal{I})_{\prec}) \bullet > \vartriangle > \;=\; (\mathcal{I} \vartriangle \geqslant \cup \geqslant \vartriangle \mathcal{I}) \bullet > \\
= & \qquad \{ \text{ cupjunctivity of composition } \} \\
 & (\mathcal{I} \vartriangle \geqslant)_{\prec} \bullet > \vartriangle > \cup (\geqslant \vartriangle \mathcal{I})_{\prec} \bullet > \vartriangle > \;=\; \mathcal{I} \vartriangle \geqslant \bullet > \cup \geqslant \vartriangle \mathcal{I} \bullet > \\
\Leftarrow & \qquad \{ \text{ Leibniz } \} \\
 & (\mathcal{I} \vartriangle \geqslant)_{\prec} \bullet > \vartriangle > \;=\; \mathcal{I} \vartriangle \geqslant \bullet > \;\wedge\; (\geqslant \vartriangle \mathcal{I})_{\prec} \bullet > \vartriangle > \;=\; \geqslant \vartriangle \mathcal{I} \bullet >
\end{array}$$

We continue with the first disjunct. By mutual inclusion:

$$\begin{array}{ll}
 & (\mathcal{I} \vartriangle \geqslant)_{\prec} \bullet > \vartriangle > \\
\subseteq & \qquad \{ \text{ monotonicity } \} \\
 & (\mathcal{I} \vartriangle \geqslant)_{\prec} \bullet > \vartriangle \top \\
= & \qquad \{ \text{ Corollary 3.24b } \} \\
 & (\mathcal{I} \vartriangle \geqslant)_{\prec} \bullet \mathcal{I} \vartriangle \top \bullet > \\
= & \qquad \{ \; (\mathcal{I} \vartriangle r)_{\prec} \bullet \mathcal{I} \vartriangle \top = \mathcal{I} \vartriangle r \; \} \\
 & \mathcal{I} \vartriangle \geqslant \bullet > \\
= & \qquad \{ \text{ Theorem 3.30b } \} \\
 & (\mathcal{I} \vartriangle \geqslant)_{\prec} \bullet \mathcal{I} \vartriangle \geqslant \bullet > \\
\subseteq & \qquad \{ \; r \vartriangle s \bullet u \subseteq (r \bullet u) \vartriangle (s \bullet u) \; \}
\end{array}$$

$$
\begin{array}{ll}
 & (\mathcal{I} \blacktriangle \geqslant)^{\prec} \bullet > \blacktriangle (\geqslant \bullet >) \\
= & \quad \{ \ \geqslant \bullet > = > \ \} \\
 & (\mathcal{I} \blacktriangle \geqslant)^{\prec} \bullet > \blacktriangle >
\end{array}
$$

□

Now the fact that $\downarrow \bullet \alpha \blacktriangle \beta$ is monotonic is easily proved:

Proposition A.7

$$\downarrow \bullet \alpha \blacktriangle \beta \bullet < \ = \ < \bullet \downarrow \bullet \alpha \blacktriangle \beta$$

Proof:
Because $\downarrow \bullet \alpha \blacktriangle \beta$ is an isomorphism on $\mathcal{T}$ (which is totally ordered by $\leqslant$) we only have to prove one inclusion:

$$
\begin{array}{ll}
 & \downarrow \bullet \alpha \blacktriangle \beta \bullet < \\
\subseteq & \quad \{ \ r \blacktriangle s \bullet u \ \subseteq \ (r \bullet u) \blacktriangle (s \bullet u) \ \} \\
 & \downarrow \bullet (\alpha \bullet <) \blacktriangle (\beta \bullet <) \\
= & \quad \{ \ \alpha \text{ and } \beta \text{ are monotonic } \} \\
 & \downarrow \bullet (< \bullet \alpha) \blacktriangle (< \bullet \beta) \\
= & \quad \{ \ \text{Product-split fusion 3.12a } \} \\
 & \downarrow \bullet < \times < \bullet \alpha \blacktriangle \beta \\
= & \quad \{ \ \text{Proposition A.6 } \} \\
 & < \bullet \downarrow \bullet \alpha \blacktriangle \beta
\end{array}
$$

□

The conclusion of the calculations is that $\downarrow \bullet \alpha \blacktriangle \beta$ is an archimedean function whenever α and β are:

Proposition A.8

$$arch.(\downarrow \bullet \alpha \blacktriangle \beta)$$

Proof:
Propositions A.3, A.5 and A.7
□

A.4 Corollaries

So much for this *arch* property of $\downarrow \bullet \alpha \blacktriangle \beta$. The discussion is completed by the following propositions:

Proposition A.9

$$^{<}(\downarrow \bullet \alpha \blacktriangle \beta \bullet t) = {}^{<}(\alpha \bullet t) \sqcup {}^{<}(\beta \bullet t)$$

Proof:

$$
\begin{array}{ll}
 & {}^{<}(\alpha \bullet t) \sqcup {}^{<}(\beta \bullet t) \\
= & \quad \{ \ \alpha \in \mathcal{T} \leftrightarrow \mathcal{T} \text{: } (\alpha \bullet t)_{\prec} \text{ is moment; Proposition 6.13a } \} \\
 & {}^{<}(\alpha \bullet t) \circ {}^{<}(\beta \bullet t) \\
= & \quad \{ \text{ Characterisation 6.10 } \} \\
 & {}^{\circ}((< \bullet \alpha \bullet t)_{\prec}) \circ {}^{\circ}((< \bullet \beta \bullet t)_{\prec}) \\
= & \quad \{ \text{ Proposition 6.8 } \} \\
 & {}^{\circ}((< \bullet \alpha \bullet t)_{\prec} \bullet (< \bullet \beta \bullet t)_{\prec}) \\
= & \quad \{ \text{ Corollary 3.27b } \} \\
 & {}^{\circ}((< \bullet \alpha \bullet t)_{\prec} \cap (< \bullet \beta \bullet t)_{\prec}) \\
= & \quad \{ \text{ Definition 3.29 } \} \\
 & {}^{\circ}(< \bullet \alpha \bullet t \bullet \top \cap < \bullet \beta \bullet t \bullet \top \cap \mathcal{I}) \\
= & \quad \{ \text{ Proposition 4.10c: } t \bullet \top \text{ is a function; Theorem 3.35a } \} \\
 & {}^{\circ}((< \bullet \alpha \cap < \bullet \beta) \bullet t \bullet \top \cap \mathcal{I}) \\
= & \quad \{ \text{ Definition 3.29 } \} \\
 & {}^{\circ}(((< \bullet \alpha \cap < \bullet \beta) \bullet t)_{\prec}) \\
= & \quad \{ \text{ below } \} \\
 & {}^{\circ}((< \bullet \downarrow \bullet \alpha \blacktriangle \beta \bullet t)_{\prec}) \\
= & \quad \{ \text{ Characterisation 6.10 } \} \\
 & {}^{<}(\downarrow \bullet \alpha \blacktriangle \beta \bullet t)
\end{array}
$$

Below:

$$
\begin{array}{ll}
 & < \bullet \downarrow \bullet r \blacktriangle s \\
= & \quad \{ \text{ Proposition A.6 } \} \\
 & \downarrow \bullet < \times < \bullet r \blacktriangle s \\
= & \quad \{ \text{ Parallel-split fusion 3.12a } \} \\
 & \downarrow \bullet (< \bullet r) \blacktriangle (< \bullet s) \\
= & \quad \{ \text{ Proposition A.2 } \} \\
 & (< \bullet r \cap \leqslant \bullet < \bullet s) \cup (\leqslant \bullet < \bullet r \cap < \bullet s) \\
= & \quad \{ \ \leqslant \bullet < = < \ \} \\
 & (< \bullet r \cap < \bullet s) \cup (< \bullet r \cap < \bullet s) \\
= & \quad \{ \text{ idempotency of cup } \} \\
 & < \bullet r \cap < \bullet s
\end{array}
$$

□

With the straightforward corollary:

Proposition A.10

a. ${}^{<}(\downarrow \bullet \alpha \blacktriangle \beta \bullet t) \sqsupseteq {}^{<}(\alpha \bullet t)$

b. ${}^{<}(\downarrow \bullet \alpha \blacktriangle \beta \bullet t) \sqsupseteq {}^{<}(\beta \bullet t)$

□

This concludes this appendix on $\downarrow$.

Appendix B

Axiom of Choice applied

The Axiom of Choice for Chronicles 4.20 is used to prove Proposition 10.14b of *cut*:

$$r^\circ \circ cut \circ (r^\circ)_> \sqsubseteq cut \circ r^\circ \;\Leftarrow\; r \bullet wipe \bullet r_\succ \subseteq wipe \bullet r$$

Proof:
Using the definition of *cut*, Characterisation 10.11, the consequent follows by cupjunctivity from:

$$r^\circ \circ ({}^<t \sqcap {}^\geqslant t \circ wipe^\circ) \circ (r^\circ)_> \;\sqsubseteq\; ({}^<t \sqcap {}^\geqslant t \circ wipe^\circ) \circ r^\circ$$

The latter inclusion is shown in the model: for all f and g,

$$\begin{array}{ll}
 & f \langle r^\circ \circ ({}^<t \sqcap {}^\geqslant t \circ wipe^\circ) \circ (r^\circ)_> \rangle\, g \\
= & \quad \{ \text{ definitions in the model } \} \\
 & \exists (\, h, h' \;::\; f \subseteq r \bullet h \;\wedge\; h = g \bullet (<\bullet t)_\prec \cup wipe \bullet g \bullet (\geqslant \bullet t)_\prec \;\wedge\; h' \subseteq r \bullet g \,) \\
\Rightarrow & \quad \{ \text{ monotonicity; } \exists (\, h' \;::\; h' \subseteq r \bullet g \,) \;\Rightarrow\; g = r_\succ \bullet g \;\} \\
 & f \subseteq r \bullet g \bullet (<\bullet t)_\prec \cup r \bullet wipe \bullet g \bullet (\geqslant \bullet t)_\prec \;\wedge\; g = r_\succ \bullet g \\
\Rightarrow & \quad \{ \text{ define } rr = r \bullet g \cap (f \bullet (<\bullet t)_\prec \cup wipe^{-1} \bullet f \bullet (\geqslant \bullet t)_\prec); \text{ below } \} \\
 & rr_\succ = \mathcal{T} \;\wedge\; sp.rr \text{ is well-ordered} \;\wedge\; unar.rr \\
\Rightarrow & \quad \{ \text{ Axiom of Choice for Chronicles 4.20 } \} \\
 & \exists (\, h \;::\; h \subseteq rr \,) \\
= & \quad \{ \text{ definition } rr; \text{ property cap } \} \\
 & \exists (\, h \;::\; h \subseteq f \bullet (<\bullet t)_\prec \cup wipe^{-1} \bullet f \bullet (\geqslant \bullet t)_\prec \;\wedge\; h \subseteq r \bullet g \,) \\
\Rightarrow & \quad \{ \text{ Proposition 6.9a; identities } \} \\
 & \exists (\, h \;::\; f \supseteq h \bullet (<\bullet t)_\prec \;\wedge\; wipe^{-1} \bullet f \supseteq h \bullet (\geqslant \bullet t)_\prec \;\wedge\; h \subseteq r \bullet g \,) \\
= & \quad \{ \; wipe \text{ is a total function } \} \\
 & \exists (\, h \;::\; f \supseteq h \bullet (<\bullet t)_\prec \;\wedge\; f \supseteq wipe \bullet h \bullet (\geqslant \bullet t)_\prec \;\wedge\; h \subseteq r \bullet g \,) \\
= & \quad \{ \text{ property cup } \} \\
 & \exists (\, h \;::\; f \supseteq h \bullet (<\bullet t)_\prec \cup wipe \bullet h \bullet (\geqslant \bullet t)_\prec \;\wedge\; h \subseteq r \bullet g \,) \\
= & \quad \{ \text{ Proposition 6.9b: } h \bullet (<\bullet t)_\prec \cup wipe \bullet h \bullet (\geqslant \bullet t)_\prec \text{ is total } \} \\
 & \exists (\, h \;::\; f = h \bullet (<\bullet t)_\prec \cup wipe \bullet h \bullet (\geqslant \bullet t)_\prec \;\wedge\; h \subseteq r \bullet g \,) \\
= & \quad \{ \text{ definitions in the model } \} \\
 & f \langle ({}^<t \sqcap {}^\geqslant t \circ wipe^\circ) \circ r^\circ \rangle\, g
\end{array}$$

We skipped a step which led us to the conditions of the Axiom of Choice for Chronicles 4.20. Let $a = (< \bullet t)_{\prec}$. It follows by Proposition 6.9 that $(\geqslant \bullet t)_{\prec}$ is the complement of a (with respect to $\mathcal{T}$); this complement is denoted by $\tilde{a}$. The conditions that we have to check are:

$$
\begin{array}{ll}
 & rr_{\succ} = \mathcal{T} \\
\wedge & \\
 & sp.rr \text{ is well-ordered} \\
\wedge & \\
 & unar.rr
\end{array}
$$

This has to follow from the assumptions:

$$
\begin{array}{ll}
 & r \bullet wipe \bullet r_{\succ} \subseteq wipe \bullet r \\
\wedge & \\
 & f \subseteq r \bullet g \bullet (< \bullet t)_{\prec} \cup r \bullet wipe \bullet g \bullet (\geqslant \bullet t)_{\prec} \\
\wedge & \\
 & g = r_{\succ} \bullet g
\end{array}
$$

We start with $rr_{\succ} = \mathcal{T}$. First:

$$
\begin{array}{ll}
 & f \bullet a \\
\subseteq & \quad \{ \text{ assumption on } f \ \} \\
 & (r \bullet g \bullet a \cup r \bullet wipe \bullet g \bullet \tilde{a}) \ \bullet \ a \\
= & \quad \{ \text{ cupjunctivity; } \tilde{a} \bullet a = \bot \ \} \\
 & r \bullet g \bullet a \\
\subseteq & \quad \{ \text{ identity } \} \\
 & r \bullet g
\end{array}
$$

And second:

$$
\begin{array}{ll}
 & f \bullet \tilde{a} \\
\subseteq & \quad \{ \text{ assumption on } f \text{; cupjunctivity; } \tilde{a} \bullet a = \bot \ \} \\
 & r \bullet wipe \bullet g \bullet \tilde{a} \\
\subseteq & \quad \{ \text{ identity } \} \\
 & r \bullet wipe \bullet g \\
= & \quad \{ \text{ assumption on } g \ \} \\
 & r \bullet wipe \bullet r_{\succ} \bullet g \\
\subseteq & \quad \{ \text{ assumption on } r \ \} \\
 & wipe \bullet r \bullet g
\end{array}
$$

The totality of rr follows:

$$
\begin{array}{ll}
 & rr_{\succ} \\
= & \quad \{ \text{ definition } rr \ \} \\
 & (r \bullet g \ \cap \ (f \bullet a \cup wipe^{-1} \bullet f \bullet \tilde{a}))_{\succ} \\
= & \quad \{ \text{ cupjunctivity } \} \\
 & (r \bullet g \cap f \bullet a)_{\succ} \ \cup \ (r \bullet g \cap wipe^{-1} \bullet f \bullet \tilde{a})_{\succ} \\
= & \quad \{ \text{ domains: } (s \cap u \bullet v)_{\succ} \ = \ (u^{-1} \bullet s \cap v)_{\succ} \ \}
\end{array}
$$

$$
\begin{array}{ll}
 & (r \bullet g \cap f \bullet a)\rhd \ \cup \ (wipe \bullet r \bullet g \cap f \bullet \tilde{a})\rhd \\
= & \qquad \{ \text{ first and second result above } \} \\
 & (f \bullet a)\rhd \cup (f \bullet \tilde{a})\rhd \\
= & \qquad \{ \text{ cupjunctivity } \} \\
 & (f \bullet (a \cup \tilde{a}))\rhd \\
= & \qquad \{ \ a \cup \tilde{a} = \mathcal{T} \ \} \\
 & (f \bullet \mathcal{T})\rhd \\
= & \qquad \{ \text{ chronicles are total on } \mathcal{T} \ \} \\
 & \mathcal{T}
\end{array}
$$

We continue with the proof that the support of rr is well-ordered. This follows from the fact that $sp.rr$ is included in the support of g. First, we show $sm \bullet r \subseteq r \bullet sm$, which follows from the assumption on r and the fact that sm and $wipe\lhd$ are complementary domains:

$$
\begin{array}{ll}
 & sm \bullet r \subseteq r \bullet sm \\
= & \qquad \{ \ sm \cup wipe\lhd = \mathcal{I} \text{: cupjunctivity } \} \\
 & sm \bullet r \bullet sm \ \cup \ sm \bullet r \bullet wipe\lhd \ \subseteq \ r \bullet sm \\
= & \qquad \{ \ sm \cap wipe\lhd = \bot \text{: } sm \bullet wipe \bullet r = \bot \ \} \\
 & sm \bullet r \bullet sm \ \cup \ sm \bullet r \bullet wipe\lhd \ \subseteq \ r \bullet sm \ \cup \ sm \bullet wipe \bullet r \\
\Leftarrow & \qquad \{ \ sm \subseteq \mathcal{I} \text{: monotonicity } \} \\
 & r \bullet wipe\lhd \subseteq wipe \bullet r \\
= & \qquad \{ \text{ domains } \} \\
 & r \bullet wipe\lhd \bullet r\rhd \subseteq wipe \bullet r \\
\Leftarrow & \qquad \{ \text{ Theorem 4.17: } wipe \text{ is an idempotent function, so } wipe\lhd \subseteq wipe \ \} \\
 & r \bullet wipe \bullet r\rhd \subseteq wipe \bullet r \\
= & \qquad \{ \text{ assumption on } r \ \} \\
 & True
\end{array}
$$

The claimed inclusion of the supports follows:

$$
\begin{array}{ll}
 & sp.rr \\
= & \qquad \{ \ rr\rhd = \mathcal{T} \text{: Characterisation 4.11; definition } rr \ \} \\
 & (sm \ \bullet \ (r \bullet g \ \cap \ (f \bullet a \ \cup \ wipe^{-1} \bullet f \bullet \tilde{a})))\rhd \\
\subseteq & \qquad \{ \text{ monotonicity } \} \\
 & (sm \bullet r \bullet g)\rhd \\
\subseteq & \qquad \{ \text{ above: monotonicity } \} \\
 & (r \bullet sm \bullet g)\rhd \\
\subseteq & \qquad \{ \text{ domains: } (s \bullet u)\rhd \subseteq u\rhd \ \} \\
 & (sm \bullet g)\rhd \\
= & \qquad \{ \text{ Characterisation 4.11 } \} \\
 & sp.g
\end{array}
$$

Finally, we show that rr has a unique arity; in fact, its arity is the same as that of f:

$$
\begin{array}{ll}
 & rr \bullet \top \bullet rr^{-1} \\
= & \qquad \{ \text{ definition } rr \ \}
\end{array}
$$

$$
\begin{array}{ll}
 & (r \bullet g \cap (f \bullet a \cup wipe^{-1} \bullet f \bullet \tilde{a})) \bullet \top \bullet (r \bullet g \cap (f \bullet a \cup wipe^{-1} \bullet f \bullet \tilde{a}))^{-1} \\
\subseteq & \quad \{ \text{ monotonicity } \} \\
 & (f \bullet a \cup wipe^{-1} \bullet f \bullet \tilde{a}) \bullet \top \bullet (f \bullet a \cup wipe^{-1} \bullet f \bullet \tilde{a})^{-1} \\
\subseteq & \quad \{ \text{ identities: monotonicity } \} \\
 & (f \cup wipe^{-1} \bullet f) \bullet \top \bullet (f \cup wipe^{-1} \bullet f)^{-1} \\
= & \quad \{ \text{ cupjunctivity and reverse } \} \\
 & (\mathcal{I} \cup wipe^{-1}) \bullet f \bullet \top \bullet f^{-1} \bullet (\mathcal{I} \cup wipe) \\
\subseteq & \quad \{ \text{ Proposition 4.19 and Definition 4.18 } \} \\
 & (\mathcal{I} \cup wipe^{-1}) \bullet wipe^{-1} \bullet wipe \bullet (\mathcal{I} \cup wipe) \\
= & \quad \{ \text{ cupjunctivity; Theorem 4.17a } \} \\
 & wipe^{-1} \bullet wipe
\end{array}
$$

This proves by Definition 4.18 the required uniqueness of the arity of rr, and thereby the second statement of Proposition 10.14.

□

It should be remarked that Assumption 4.20 is not used in proving the first statement of Proposition 10.14.

Appendix C

Dataflow

The model on which we based our calculus of procs is rather basic. There is reason to expect that other theories for communicating processes can be expressed in terms of our model. As an example, we sketch how to make a link between our calculus of procs and the theory of dataflow networks, see for a reference Kok [Kok93]. To model dataflow networks in our calculus of procs, we have to relate several constructions such as compositions and processes in dataflow theory with constructions in the proc calculus. As a start, we first define the parallel closure of *pdly*, which corresponds to a buffered channel:

Definition C.1

$$Pdly \triangleq \mu(\, X \;::\; pdly \sqcup X \parallel X \,)$$

□

The proc *Pdly* is able to delay multiple parallel channels independently of each other. Contexts in dataflow theory are built using buffered processes and the compositions $\circ$, $\parallel$, $\triangle$ and $^{\sigma}$. A buffered process in our calculus is defined as follows:

Definition C.2

$$buff.P \triangleq Pdly \circ P \circ Pdly$$

□

In addition, the observation criterion, $=_{hist}$ is defined:

Definition C.3

a. $$\widetilde{P} \triangleq Pdly \circ Pdly^{\cup} \circ P \circ Pdly \circ Pdly^{\cup}$$

b. $$P =_{hist} Q \;\equiv\; \widetilde{P} = \widetilde{Q}$$

□

The proc $\widetilde{P}$ corresponds to the history relation of P. Therefore, the equivalence relation $=_{hist}$ is the equality on history relations.

It is known that the observation criterion $=_{hist}$ is not a congruence for all contexts, see Brock & Ackerman [BA81]: there exist procs P and Q and a context C such that:

$$P =_{hist} Q \ \wedge \ \neg(C[P] =_{hist} C[Q])$$

We say that $=_{hist}$ is not compositional. The largest congruence between procs contained in $=_{hist}$ is $\lambda(\, P, Q \ :: \ \forall(\, C \ :: \ C[P] =_{hist} C[Q]\,)\,)$:

Definition C.4

$$P \approx_{hist} Q \ \triangleq \ \forall(\, C \ :: \ C[P] =_{hist} C[Q]\,)$$

□

The question raised now is: which procs are equal in the sense of $\approx_{hist}$? To answer this question we record an important conjecture: the congruence $\approx_{hist}$ can be rewritten using the procs $\widehat{P}$:

Characterisation C.5

a. $$trace.P \triangleq \sqcup(\, \varsigma \ : \ oiso.\varsigma \ : \ {}^{\circ}\varsigma \circ P \circ ({}^{\circ}\varsigma)^{\cup}\,)$$

b. $$\widehat{P} \triangleq trace.(buff.P)$$

□

Procs satisfying $P = trace.P$ abstract from exact timing. For these procs, only the *order* of (input and output) messages is important. Such procs can be characterised by a set of traces. Therefore, the proc $\widehat{P}$ corresponds to the description of P as a set of traces which are closed under buffering (the so-called *Buffering Condition*). The conjecture reads:

Conjecture C.6

$$P \approx_{hist} Q \ \equiv \ \widehat{P} = \widehat{Q}$$

□

This is suggested by corresponding theorems in dataflow theory: it is our version of the Full Abstraction Theorem. Future research has to check this claim.

Samenvatting

Informele introductie

In deze moderne tijd wordt het kunnen localiseren en verkrijgen van de juiste informatie steeds belangrijker. Wil je goede service leveren of de concurrent aftroeven, dan zul je slimmer moeten zijn, en zorgen dat je de belangrijkste informatie (snel) tot je beschikking hebt. Met name de elektronische informatieverwerking neemt een belangrijke plaats in; denk hierbij aan reserveringssystemen voor vliegtuigen en aan nieuwsnetwerken. Dit zijn grote, vaak wereldwijde netwerken waar honderden of zelfs duizenden computers aan gekoppeld zijn. Deze computers verzenden, ontvangen en presenteren boodschappen.
Om het informatieverkeer over zo'n netwerk ordelijk te laten verlopen zijn er regels nodig: het schaarse produkt (toegang tot de diensten van het netwerk) moet 'eerlijk' verdeeld worden. Het mag bijvoorbeeld niet mogelijk zijn dat dezelfde vliegtuigstoel gelijktijdig vanuit twee plaatsen wordt gereserveerd. Maar er mag ook geen "na u – na u" situatie ontstaan waarbij twee klanten van het netwerk eindeloos op elkaar blijven wachten. Verder zal er gezorgd moeten worden voor adequate oplossingen wanneer er problemen optreden in het netwerk, zoals het uitvallen van een verbinding. Daarom zal een netwerk, ongeacht de grootte, moeten werken volgens regels welke zijn vastgelegd in computerprogramma's: zogenaamde netwerkprotocollen. Omdat we te maken hebben met meerdere computers in een netwerk, zijn deze protocollen verdeeld (gedistribueerd) over de verschillende computers, en spreken we van gedistribueerde protocollen. De doelstelling van het onderzoek dat in dit proefschrift wordt gepresenteerd is het ontwikkelen van een rekenmethode (calculus) om dit soort gedistribueerde protocollen op elegante en correcte wijze te kunnen ontwerpen.

Waarom een formele rekenmethode? Het menselijk verstand is in het algemeen niet in staat om de complexiteit die ontstaat bij het ontwerpen van programma's te overzien. Daarom moeten er hulpmiddelen geleverd worden om programma's te kunnen ontwerpen; in het bijzonder als het gedistribueerde programma's betreft. Deze hulpmiddelen kunnen de vorm krijgen van een verzameling rekenregels om uit een formele beschrijving (specificatie) een programma af te kunnen leiden. Een complete verzameling van rekenregels en heuristieken vormt de uiteindelijke calculus.
Een groot probleem is dat voor het berekenen van een gedistribueerd programma vaak rekening moet worden gehouden met veel verschillende situaties. De reden is simpel: een groot systeem van computers die met elkaar willen communiceren kent vele verschillende toestanden. Om het hoofd te kunnen bieden aan al die situaties is het belangrijk dat

irrelevante details van het systeem achterwege gelaten worden. Daarom willen we niet redeneren over het ontvangen of verzenden van één boodschap, zelfs niet over het ontvangen of verzenden van een complete stroom boodschappen, maar over het complete gedrag van een programma in elke willekeurige omgeving. De omgeving wordt bepaald door de andere programma's in het netwerk.

Vaak kan het gedrag van een programma vastgelegd worden in een verzameling rekenregels door de interactie met andere (bekende) programma's te geven. Een simpel voorbeeld daarvan is het karakteriseren van het programma I dat iedere boodschap a binnengekomen op tijdstip t meteen weer doorstuurt:

Voor alle boodschappen a en tijdstippen t: $I.a_t = a_t$

Hier worden dus afzonderlijke boodschappen genoemd. Beter zou het zijn (maar nog niet goed genoeg) om I te karakteriseren door te zeggen dat iedere complete invoerstroom f van boodschappen ook de uitvoer is:

Voor alle stromen van boodschappen f: $I.f = f$

In dit proefschrift wordt I vastgelegd door het gedrag te geven in iedere willekeurige omgeving:

Voor alle programma's P: $I \circ P = P$ en $P = P \circ I$

Deze laatste formulering geeft aan dat I het gedrag van ieder ander willekeurig programma P ongemoeid laat.

Omdat het gedrag van een groot computernetwerk niet volledig te controleren is (bijvoorbeeld onvoorspelbaar menselijk gedrag kan een grote rol spelen), moeten we in de calculus rekening kunnen houden met onbepaald gedrag (non-determinisme). Een van de belangrijkste onbepaalde factoren is de tijdsduur: vaak is wel duidelijk *welke* boodschap verstuurd is over een verbinding, maar is het onbepaald hoe *lang* die boodschap er over zal doen. Een andere onbepaalde factor is dat op de meest willekeurige momenten plotseling een fout ergens in het systeem kan optreden. Om over dit soort onbepaald gedrag te kunnen redeneren, is de calculus in dit proefschrift gebouwd op een zogenaamde relationele calculus. Deze relationele calculus kan goed overweg met non-determinisme.

Kort overzicht

Het proefschrift bestaat uit vijf delen. Elk deel breidt de calculus verder uit. Tot slot wordt de calculus gebruikt voor het ontwerpen van een simpel, gedistribueerd protocol voor het correct verzenden door een zender van boodschappen over onbetrouwbare verbindingen naar een ontvanger.

In Deel I wordt eerst aangegeven hoe we gedistribueerde systemen willen beschrijven als invoer/uitvoer-relaties. Vervolgens wordt de relationele calculus gepresenteerd. In deze relationele calculus wordt, ter verkrijging van een relationele calculus voor het ontwerpen van gedistribueerde systemen, meer structuur aangebracht door de constructie van

de feedback loop te definiëren. Er wordt aangetoond dat feedback een aantal 'ongezonde' eigenschappen heeft. Een groot deel van het proefschrift (in het bijzonder Deel III) zal zich wijden aan het oplossen van deze problemen. Tenslotte wordt het model van de relationele calculus, waarin veel voorbereidende berekeningen zullen worden uitgevoerd, vastgelegd. De elementen van dit model, de zogenaamde chronicles, beschrijven complete stromen boodschappen.

Deel II introduceert twee constructies om chronicles te kunnen manipuleren. De functie postcompose kan worden gebruikt om functies die werken op afzonderlijke boodschappen om te zetten in processen die werken op complete stromen van boodschappen. Zo kunnen we met de optelling + voor natuurlijke getallen een eenvoudig proces beschrijven dat een complete stroom van paren van natuurlijke getallen omzet in de stroom van sommen van de paren. Enkele basisprocessen worden met behulp van postcompose gedefinieerd en onderzocht. De functie precompose kan veranderingen aanbrengen in de tijd. Omdat we in de calculus niet willen redeneren over exacte tijd of tijdsduur, is de functie precompose geen onderdeel van de uiteindelijke calculus, maar slechts een hulpmiddel om bijvoorbeeld een (onbepaalde) vertraging te beschrijven.

In Deel I zijn verschillende ongezonde eigenschappen van met name de feedback gesignaleerd. Deel III presenteert daarom een nieuwe eigenschap, genaamd causaliteit. Een proces dat deze eigenschap heeft is gezond in de zin dat er geen boodschappen (van een bepaald type) geweigerd kunnen worden, en dat huidige uitvoer niet afhangt van huidige of toekomstige invoer. Dit zijn realistische eigenschappen die niet noodzakelijk gelden voor ieder proces dat uitgedrukt kan worden in de calculus. Causale processen worden uitvoerig onderzocht, met name in combinatie met feedback. Het blijkt dat causale processen in een feedback loop weer een causaal proces opleveren.

Deel IV definieert en onderzoekt, in het model, drie basisprocessen die verschuivingen kunnen aanbrengen in de tijd. Twee processen drukken de vertraging van een verbinding uit. Een derde proces heeft de mogelijkheid om de doorvoer van boodschappen af te kappen. Daarna volgen definities van nieuwe processen in termen van de drie basisprocessen. Het belangrijkste nieuwe proces is de buffer, welke gebruikt kan worden voor asynchrone communicatie.

Tenslotte wordt al het werk van de eerste vier delen samengebracht in de afleiding van een eenvoudig communicatieprotocol. Daartoe wordt in Deel V eerst getoond hoe de algemene theorie kan worden geïnstantieerd tot een meer gespecialiseerde theorie. De daarop volgende afleiding van het communicatieprotocol kenmerkt zich door het abstracte niveau waarop de verschillende bewijsverplichtingen worden afgehandeld: er wordt slechts geredeneerd over processen, niet over (stromen van) boodschappen.

Curriculum vitae

Frans Johan Rietman

26 september 1967
Geboren te Kampen.

1979 - 1985
VWO aan het Johannes Calvijn-Lyceum te Kampen.

1985 - 1986
Studie Elektrotechniek aan de HTS te Zwolle.

1986 - 1991
Studie Informatica aan de Rijksuniversiteit Groningen. Afstudeerverslag: An Aggregated Segment Sum Theorem in the Relational System.

1991 - 1995
Onderzoeker in opleiding aan de Vakgroep Informatica van de Universiteit Utrecht, in dienst van de Nederlandse Organisatie voor Wetenschappelijk Onderzoek (NWO).