Linear unary operators in process algebra

Linear unary operators in process algebra

Academisch proefschrift

ter verkrijging van de graad van doctor
aan de Universiteit van Amsterdam,
op gezag van de Rector Magnificus
prof. dr. P. W. M. de Meijer
in het openbaar te verdedigen in de Aula der Universiteit
(Oude Lutherse Kerk, ingang Singel 411, hoek Spui),
op maandag 1 juni 1992 te 10.30 uur

door Christiaan Verhoef

geboren te Kedichem

Promotor: Prof. dr. Jan Bergstra
Faculteit: Wiskunde en Informatica

Preface

THIS THESIS is devoted to the study of linear unary operators in process algebra. The idea to do systematic research on this subject came after finishing my first technical report [**43**]. In this report we proposed a solution for a problem by means of an operator introduced in an ad hoc way. The idea of solving a problem in this way is not very original. In fact, people working in the field of process algebra are used to solve problems in this manner. We decided that it was probably an interesting task to design a unifying framework for such theories. At that time we had no idea of the amount of work that accompanied this target. So it will be not very surprising that we have not fulfilled our aims yet. Nonetheless, in this thesis a lot of work has already been done. In the following section we will give an overview of what can be expected.

Overview

In this section an overview of the contents of this thesis is given. A linear unary operator is an operator that distributes over alternative composition. Its domain and its image is the sort of processes. And, of course, a linear *unary* operator takes one argument.

Chapter 1 provides a short and incomplete introduction to process algebra. However, an elaborate "list" of linear unary operators can be found in this chapter. After all, this thesis is about linear unary operators.

Chapter 2 can be seen as a case study concerning linear unary operators. Nearly all the problems that can be thought of for the introduction of an operator come into play in this chapter. A way to handle processes with parallel input is treated in this chapter. To obtain this, a linear unary operator called the register operator is proposed that can be seen as a "memory" that special atomic actions can use to read or write data. It is shown that the register operator distributes over the merge operator, which is an important result for the applications. As an application we prove a correctness theorem on palindrome recognition with the use of the register operator.

In the subsequent chapter 3 a first attempt is made in order to obtain a uniform approach for the introduction of linear unary operators. For this purpose a proof rule is proposed. In order to be able to reason in a comfortable way about linear unary operators a second proof rule is added, as well. Standard facts such as termination and elimination are presented in this chapter.

A model for the theory without abstraction is given. General theorems on linear unary operators in which the proof rules have been used extensively form an important part of this chapter. By a number of examples, originally treated in different theories, it is shown that the proposed theory is unifying since they can be handled with one theory. Finally, generalizations in order to be able to incorporate more and more linear unary operators are briefly mentioned.

In chapter 4 one of the recommendations given in chapter 3 has been worked out. In chapter 3 it was not yet possible to reason smoothly about the class of projection operators, but in chapter 4 this is solved by generalizing some aspects of chapter 3. Besides the two proof rules of chapter 3, we propose a proof rule in order to be able to give inductive proofs. The termination, elimination and the model without abstraction are not present in this chapter, but they can be imitated effortlessly with the aid of the results in chapter 3. However, a section on general theorems concerning linear unary operators is included, in which theorems on projection operators form the main part. Then a section follows in which the use of the third proof rule is shown. It is proved that a fair FIFO queue satisfies a criterion for protocol correctness. The method is applied to several alternating bit protocols with a time out that can take place at any moment.

Acknowledgements

Writing a thesis is, in general, a hard and solitary job. The present thesis is not an exception to this rule. Nevertheless, I want to emphasize that Jos Baeten and Jan Bergstra gave me the tools that made the task of writing it feasible. Jos gave me a problem which led to chapter 2 and Jan was always willing to reflect my ideas in such a clear way that, thereafter, I really understood them. These discussions, among others, led to chapters 3 and 4. Therefore, I am grateful to both of them. I also consider the pleasant and scientific atmosphere at the programming research group an important factor in the materialization of this book. So I definitely want to thank all the members of the programming research group of the University of Amsterdam!

Finally, I would like to thank Jan Bergstra for being a wonderful promotor, Jos Baeten, Johan van Benthem, Jan Friso Groote and Paul Klint for their willingness to be a member of the committee of graduation, Jos Baeten, Jan Friso Groote and Eric Nieuwland for their meticulous reading of the penultimate version of this thesis, Jan Friso Groote for recommending an epentheticum in the form of remark (3.5.28) and Edward Pijpers for designing the cover.

Contents

Note

Chapter 1 is based on [**15**]. Chapters 2, 3 and 4 have been published as technical reports of the University of Amsterdam. Their references are respectively [**43**], [**41**] and [**42**].

Chapter 1

Introduction

IN this chapter we will provide a short introduction to the basic notions that will be used in the rest of this thesis. In fact, the other chapters are based on the algebraic theory ACP_τ with projection operators. The acronym stands for the Algebra of Communicating Processes with Abstraction and the subscript τ is called Milner's silent action [**36**]. The theory ACP_τ was first studied in [**12**]. Since then many extensions of this theory have been proposed. These proposals often consist of the addition of a set of linear unary operators to the axiomatic framework ACP_τ. Since this thesis is about the systematic research on linear unary operators, we will give in this chapter an overview of a number of these operators. In addition, we will treat some extra features such as the notion of a guarded recursive specification, the recursive definition principle, the recursive specification principle and the approximation induction principle (they are necessary if we want to deal with infinite processes). Finally, we will include a section on conditional axioms.

Readers familiar with ACP_τ might consider skipping this chapter, whereas those not familiar with any kind of process algebra better can read an introductory text like [**8**], [**26**], [**27**], [**36**] or [**37**].

1.1. The signature and the axioms

In this section we will discuss the signature of ACP_τ. The processes that we will consider are capable of performing atomic actions, with the idealization that the actions are events without positive duration in time. These actions are denoted by $a, b, c, \ldots$ and the set of these actions is denoted by A. Besides atomic actions there are also two so-called special constants in ACP_τ: the special constant δ or deadlock and τ or silent action. We can compose atomic actions and special constants with several operators into composite processes. We will discuss them hereinafter. There are five binary operators (they are all infix):

merge:	$\parallel : P \times P \longrightarrow P$,
left-merge:	$\mathbin{\|\!\lfloor} : P \times P \longrightarrow P$,

$$\begin{array}{ll} \text{communication-merge:} & \mid\, : P \times P \longrightarrow P, \\ \text{sequential composition:} & \cdot : P \times P \longrightarrow P, \\ \text{alternative composition:} & + : P \times P \longrightarrow P. \end{array}$$

For the communication-merge we have also a pre-defined binary function on A_δ ($= A \cup \{\delta\}$), which is also denoted by $\mid$

$$\mid\, : A_\delta \times A_\delta \longrightarrow A_\delta.$$

In fact, the theory ACP_τ is parameterized with a set of atomic actions and this binary communication function. Now we will discuss the unary operators. For each subset $H \subseteq A$ we have an encapsulation operator and for each $I \subseteq A$ an abstraction operator. We will call the set H the encapsulation set and for I we will use the name abstraction set. For our purposes we will also need projection operators so we will include them directly in the signature of ACP_τ. They can be found already in [**9**] and in process algebra we see them for the first time in [**14**].

$$\begin{array}{ll} \text{encapsulation operator:} & \partial_H : P \longrightarrow P, \\ \text{abstraction operator:} & \tau_I : P \longrightarrow P, \\ \text{projection operator:} & \pi_n : P \longrightarrow P. \end{array}$$

In fact, this concludes the discussion of the signature of ACP_τ with projection operators. But before we continue with the axioms we will informally give some intuition about the meaning of the elements of the signature. We already mentioned that the atomic actions can be seen as events without positive duration in time. The basic binary operations in process algebra are the alternative and sequential composition or sum and product. We can interpret the sum $a + b$ as the choice between a and b. We think of the product $a \cdot b$ as the process that first performs the action a and then b. This means that the alternative composition is commutative, whereas the sequential composition is not commutative. This will be reflected in the axiom system. We will discuss the special constants. The process $a \cdot b$ performs two actions and then terminates successfully, whereas the process $a \cdot b \cdot \delta$ will terminate unsuccessfully: after the execution of the a and b the process wants to continue with another proper action but it cannot. So we can see δ as inaction. Now we will discuss the silent step τ. The process $a \cdot \tau \cdot b$ performs an a then does something that we can not observe and then it will execute b and terminates successfully. We will see in the axioms that this process will be the same as $a \cdot b$.

Now we will discuss the other three binary operators. If x and y are processes, their so-called parallel composition or merge $x \parallel y$ is the process that first chooses whether to do a step in x or in y or a communication and proceeds as the parallel composition of the remainders of x and y. For describing the operator $\parallel$ we will use auxiliary binary operators $⌊\!⌊$ and $\mid$ with the interpretation that $x ⌊\!⌊ y$ is like $x \parallel y$ albeit that the initial step is performed by x and for $x \mid y$ yields that the initial step is a communication. For instance, if we know that for the pre-defined communication function we have $a \mid b = c$ we will have $a^2 \parallel b = a \cdot (a \parallel b) + b \cdot a^2 + c \cdot a$, in which a^2 is an abbreviation for $a \cdot a$.

A1	$x + y = y + x$	$x \cdot \tau = x$	T1
A2	$x + (y + z) = (x + y) + z$	$\tau \cdot x + x = x$	T2
A3	$x + x = x$	$a(\tau \cdot x + y) = a \cdot (\tau \cdot x + y) + a \cdot x$	T3
A4	$(x + y) \cdot z = x \cdot z + y \cdot z$		
A5	$(x \cdot y) \cdot z = x \cdot (y \cdot z)$	$\pi_n(a) = a$	PR1
A6	$x + \delta = x$	$\pi_1(a \cdot x) = a$	PR2
A7	$\delta \cdot x = \delta$	$\pi_{n+1}(a \cdot x) = a \cdot \pi_n(x)$	PR3
		$\pi_n(x + y) = \pi_n(x) + \pi_n(y)$	PR4
C1	$a \mid b = b \mid a$		
C2	$(a \mid b) \mid c = a \mid (b \mid c)$		
C3	$\delta \mid a = \delta$		
CM1	$x \parallel y = x \mathbin{\lfloor\!\lfloor} y + y \mathbin{\lfloor\!\lfloor} x + x \mid y$		
CM2	$a \mathbin{\lfloor\!\lfloor} x = a \cdot x$	$\tau \mathbin{\lfloor\!\lfloor} x = \tau \cdot x$	TM1
CM3	$(a \cdot x) \mathbin{\lfloor\!\lfloor} y = a \cdot (x \parallel y)$	$(\tau \cdot x) \mathbin{\lfloor\!\lfloor} y = \tau \cdot (x \parallel y)$	TM2
CM4	$(x + y) \mathbin{\lfloor\!\lfloor} z = x \mathbin{\lfloor\!\lfloor} z + y \mathbin{\lfloor\!\lfloor} z$	$\tau \mid x = \delta$	TC1
CM5	$(a \cdot x) \mid b = (a \mid b) \cdot x$	$x \mid \tau = \delta$	TC2
CM6	$a \mid (b \cdot x) = (a \mid b) \cdot x$	$(\tau \cdot x) \mid y = x \mid y$	TC3
CM7	$(a \cdot x) \mid (b \cdot y) = (a \mid b) \cdot (x \parallel y)$	$x \mid (\tau \cdot y) = x \mid y$	TC4
CM8	$(x + y) \mid z = x \mid z + y \mid z$		
CM9	$x \mid (y + z) = x \mid y + x \mid z$	$\partial_H(\tau) = \tau$	DT
		$\tau_I(\tau) = \tau$	TI1
D1	$\partial_H(a) = a, \quad \text{if } a \notin H$	$\tau_I(a) = a, \quad \text{if } a \notin I$	TI2
D2	$\partial_H(a) = \delta, \quad \text{if } a \in H$	$\tau_I(a) = \delta, \quad \text{if } a \in I$	TI3
D3	$\partial_H(x \cdot y) = \partial_H(x) \cdot \partial_H(y)$	$\tau_I(x \cdot y) = \tau_I(x) \cdot \tau_I(y)$	TI4
D4	$\partial_H(x + y) = \partial_H(x) + \partial_H(y)$	$\tau_I(x + y) = \tau_I(x) + \tau_I(y)$	TI5

Table 1.1. ACP_τ.

Now we will discuss the unary operators. In ACP_τ we have two of them:

the encapsulation and the abstraction operator. The encapsulation operator is an operator that renames the atomic actions listed in its encapsulation set H into the special constant δ. It is needed for the removal of unsuccessful attempts at communication. The encapsulation set is the set of such attempts. The abstraction operator is also a renaming operator, albeit that the actions in the abstraction set will be renamed into the silent action τ. This operator is needed to hide internal communication. Now we will discuss the projections. The nth projection $\pi_n(x)$ of a process x is the process x but cut off at level n. So $\pi_2(a \cdot b \cdot c) = a \cdot b$. But $\pi_2(a \cdot \tau \cdot c) = a \cdot c$. We see that τ has no depth. This is because of the axioms that we have for τ.

We will enumerate the axioms of ACP_τ with projection operators in table 1.1; see page 11. In this table we will use the following notational conventions: a, b and c are atomic actions or δ (we abbreviate $A_\delta = A \cup \{\delta\}$); x, y and z are processes; $H, I \subseteq A$ and $n \geq 1$.

1.2. Additional proof rules

In this section we will treat some additional proof rules with accompanying definitions. They are the recursive definition principle *RDP*, the recursive specification principle *RSP*, the approximation induction principle *AIP* and Koomen's fair abstraction rule *KFAR*. The principles *RDP*, *RSP* and *AIP* have been studied in [**17**]. The first two are an indispensable tool in algebraic verification techniques (see, for instance, [**3**] and [**38**]). The principle *AIP* has mainly been used to extend results on closed terms to a bigger class of terms. The proof rule *KFAR* has been introduced in process algebra for the algebraic verification of communication protocols [**17**]. This rule originated from a formula manipulation package that has been developed in [**31**].

Before we are in a position to formulate these principles we will need some preliminary definitions. We will need the notion of a (guarded) recursive specification. These notions are taken from [**16**], although the notion of a guard already can be found in [**14**] in process algebra. We will begin with the definition of a recursive specification.

Definition (1.2.1) *Recursive Specifications*

Let X be a set of variables. A recursive specification E with variable set X over ACP_τ is a system of recursion equations with variables in X:

$$E = \{x = t_x(X) : x \in X\}.$$

For all $x \in X$, we have that $t_x(X)$ is an ACP_τ-term with variables in X.

Definition (1.2.2)

Let t be a term over ACP_τ without the abstraction operator. Suppose that in t a variable x occurs. We will call an occurrence of x in t guarded if t has a subterm of the form $a \cdot s$, in which a is an atomic action and s is a term over ACP_τ, which contains this occurrence of x. Otherwise we will call the occurrence of x in t unguarded. We will call an ACP_τ-term t without the abstraction operator guarded if all occurrences of all variables in t are guarded. Let $E = \{x = t_x(X) : x \in X\}$ be an abstraction-free recursive specification. We will call E a guarded recursive specification if we can rewrite it to a recursive specification E' with the aid of the axioms and/or the aid of the specification E itself in which all right-hand sides of the recursion equations of E' are guarded. We will call E' a completely guarded recursive specification.

Definition (1.2.3)

Let $E = \{x = t_x(X) : x \in X\}$ be a recursive specification. A solution for E is a vector $p = (p_x)_{x \in X}$, with p_x a process, such that for all $x \in X$ the following expressions are true statements: $p_x = t_x(p)$, in which $t_x(p)$ is shorthand for: substitute for each occurrence of an element $x \in X$ in $t_x(X)$ the process p_x. We say that two solutions are equal if the components of the vectors are equal: $(p_x)_{x \in X} = (q_x)_{x \in X}$ if and only if we have for each $x \in X$ that $p_x = q_x$.

Principle (1.2.4) *The Recursive Definition Principle*

Let E be a guarded recursive specification without the abstraction operator. Then there is a solution for E. We call this principle *RDP*. It is also called RDP^-.

Principle (1.2.5) *The Recursive Specification Principle*

Let E be an abstraction-free guarded recursive specification. Then there is at most one solution for E. We call this principle *RSP*.

At this point we want to introduce the approximation induction principle. This definition can be found in [**17**].

Principle (1.2.6) *The Approximation Induction Principle*

Let x, y be ACP_τ-terms. If we have for all $n \geq 1$ that $\pi_n(x) = \pi_n(y)$, then we have $x = y$. We will abbreviate this principle by *AIP*.

Principle (1.2.7) *Koomen's Fair Abstraction Rule*

Let $x_1, \ldots, x_n$ and $y_1, \ldots, y_n$ be processes. Let $I \subseteq A$ and suppose that we have the following identities for these processes:

$$x_1 = i_1 \cdot x_2 + y_1,$$

$$x_2 = i_2 \cdot x_3 + y_2,$$
$$\vdots$$
$$x_{n-1} = i_{n-1} \cdot x_n + y_{n-1},$$
$$x_n = i_n \cdot x_1 + y_n,$$

with the following assumptions on the i_j, $(1 \leq j \leq n)$:

$$\{\tau\} \neq \{i_1, \ldots, i_n\} \subseteq I \cup \{\tau\},$$

then we have:

$$\tau_I(x) = \tau \cdot \big(\tau_I(y_1) + \tau_I(y_2) + \cdots + \tau_I(y_n)\big).$$

We will refer to this principle by the abbreviation $KFAR_n$. If $n = 1$ we will also use the acronym *KFAR*.

Remark (1.2.8)

If we use the combination of *KFAR* and *AIP* we will need a more restrictive form of *AIP*. See for details section 3.2.

1.3. Linear unary operators

In this section we will give an overview of the most important linear unary operators that have been introduced in process algebra. Linear stands for the fact that these operators distribute over the alternative composition. So a linear unary operator is just a unary operator that respects the alternative composition. In the signature of ACP_τ we have already seen two important linear unary operators: the encapsulation operator and the abstraction operator. The linearity of these operators is expressed in the axioms D4 and TI4; see table 1.1. We have also seen the projection operators. Strictly speaking they do not belong to the signature of ACP_τ but we included them since we will need them in the subsequent chapters. Moreover, the axiom systems that we will propose in other chapters will have more or less the same topology as the system in table 1.1. Because of PR4 we see that this operator is linear, as well. The operators that we will discuss in this section are the renaming operators, the simple state operator and the generalized state operator. They are important operators in process specification and verification techniques; see [**3**] for a general reference. The renaming operators can be found for the first time in process algebra in [**4**]. An improved version of these operators is introduced in [**40**]. In [**40**] the renaming operators are used for the verification of two communication protocols. In fact, renaming operators are common in most concurrency theories, see, for example, [**27**] and [**36**]. The state operator originated from [**4**] and has been used in process specification and verification in [**24**] and [**40**]. The generalized state operator is also introduced in [**4**]. The ideas behind this operator have been used in [**39**] to give a formal semantics for the parallel-oriented language POOL.

Subsequently, we will enumerate the axioms concerning the renaming operators. We will use the version that can be found in [**40**].

Definition (1.3.1) *Renaming Operators*

Let f be a function from the set A of atomic actions to the set of atomic actions joined with the set of special constants. Recall that the set of special constants $C = \{\delta, \tau\}$. Then there is a linear unary operator ρ_f replacing every occurrence of a constant $a \in A$ by $f(a)$. Such a linear unary operator is called a renaming operator. The operator ρ_f can be defined by means of the equations in table 1.2. In this table we have $a \in A$, $\gamma \in C$, id is the identity function ($id(a) = a$ for all $a \in A$) and $\circ$ is the composition of operators so $f \circ g(x) = f\big(g(x)\big)$, but we put $f \circ g(a) = g(a)$ if $g(a) \in C$.

$\rho_f(\gamma) = \gamma$	RN1
$\rho_f(a) = f(a)$	RN2
$\rho_f(x + y) = \rho_f(x) + \rho_f(y)$	RN3
$\rho_f(x \cdot y) = \rho_f(x) \cdot \rho_f(y)$	RN4
$\rho_{id}(x) = x$	RR1
$\rho_f \circ \rho_g(x) = \rho_{f \circ g}(x)$	RR2

Table 1.2. Renaming operators.

More on renaming operators can be found in [**4**].

Now we will discuss the (simple) state operator. This operator has been studied in [**4**]. In fact, the state operator is a generalized renaming operator, a renaming with a memory. This operator is useful in describing processes with an independent global state. It is useful for the translation of computer programmes (in a higher order language) into process algebra [**4**].

Definition (1.3.2) *The State Operator*

Let M and S be two given sets and let A be the set of atomic actions. Recall that we use $C = \{\delta, \tau\}$. We assume all these sets to be distinct. Suppose that we have two functions *act*, for action, and *eff*, for effect:

$$act : A \times M \times S \longrightarrow A \cup C,$$
$$eff : A \times M \times S \longrightarrow S.$$

$\lambda_s^m(\gamma) = \gamma$	SO1
$\lambda_s^m(a) = a(m,s)$	SO2
$\lambda_s^m(\tau \cdot x) = \tau \cdot \lambda_s^m(x)$	SO3
$\lambda_s^m(a \cdot x) = \lambda_s^m(a) \cdot \lambda_{s(m,a)}^m(x)$	SO4
$\lambda_s^m(x + y) = \lambda_s^m(x) + \lambda_s^m(y)$	SO5

Table 1.3. State operator.

We will write $a(m,s)$ for $act(a,m,s)$ and $s(m,a)$ for $eff(a,m,s)$.

For $m \in M$ and $s \in S$ the linear unary operators λ_s^m are introduced in table 1.3. In table 1.3 we have $s \in S$, $m \in M$, $\gamma \in C$, $a \in A$, and x, y are arbitrary processes.

Remarks (1.3.3)

The set S can be thought of as the *state space*. The set M is the set of *objects*. Usually we will talk about the state of a certain object, for instance, if m represents a computer, s describes the contents of its memory and x is the input (the programme) then $\lambda_s^m(x)$ describes what happens when x is presented to computer m in state s. The execution of an atomic action will affect a specific state, which is reflected in the axioms SO2 and SO4; see table 1.3. We see that the past tense of the atomic action a is $a(m,s)$ and that the state s has been changed to $s(m,a)$; see SO4. We also see that special constants are invariant under the application of the state operator; see SO1 (and SO3).

Now we will discuss the generalized state operator. This linear unary operator is introduced in process algebra in [**4**], in which it has been used to translate computer programmes into process algebra.

Definition (1.3.4) *The Generalized State Operator*

We define the generalized state operator as follows. The function *act* does not necessarily yield one element from $A \cup C$, but a set of elements from it. We will denote the power set of $A \cup C$ as follows: $\mathscr{P}(A \cup C)$. Let M and S be two given sets and let A be the set of atomic actions. We assume all these sets to be distinct. Suppose that we have two functions *act*, for action, and *eff*, for effect:

$$act : A \times M \times S \longrightarrow \mathscr{P}(A \cup C),$$
$$eff : A \times M \times S \times (A \cup C) \longrightarrow S.$$

We will write $a(m,s)$ for $act(a,m,s)$ and $s(m,a,b)$ for $eff(a,m,s,b)$.

In table 1.4 we have $s \in S$, $m \in M$, $\gamma \in C$, $a \in A_\delta$ and x, y are arbitrary processes. If $a(m,s) = \emptyset$ we will have $\Lambda_s^m(a) = \delta$.

$\Lambda_s^m(\gamma) = \gamma$	GS1
$\Lambda_s^m(a) = \sum_{b \in a(m,s)} b$	GS2
$\Lambda_s^m(\tau \cdot x) = \tau \cdot \Lambda_s^m(x)$	GS3
$\Lambda_s^m(a \cdot x) = \Lambda_s^m(a) \cdot \Lambda_{s(m,a,b)}^m(x)$	GS4
$\Lambda_s^m(x + y) = \Lambda_s^m(x) + \Lambda_s^m(y)$	GS5

Table 1.4. Generalized state operator.

1.4. Other operators

In this section we will mention briefly a number of unary operators that can also be found in process algebra, albeit that they do not occur in the theory as often as the operators that we discussed in section 1.3.

The first operator that we will mention is the so-called localization operator. We can think of this operator as the one that "localizes" a process to certain actions, so that, in a context, typically a merge of communicating processes, we can focus on some actions and abstract from others. This operator can be found in [**4**] and can be seen as an operator that has been developed to give a certain proof. The localization operator is linear and unary.

The next operator that we will mention is the process creation operator. It can be found in [**10**] (another approach to process creation can be found in [**2**]). The process creation operator is a linear unary operator and it is very wild: atomic actions can be mapped to open terms. This operator has been used to construct the sieve of Eratosthenes.

Finally, we will mention the priority operator. It has been introduced in process algebra in [**6**]. The priority operator θ is a non-linear unary operator. It is designed to be able to choose between two alternatives, such that the one with the highest priority is chosed. Therefore, a partial ordering on the set A of atomic actions is given. For example, if $A = \{a, b, c\}$ and we have $a < b, a < c$ and b and c are not related, we have

$$\begin{aligned} \theta(a + b) &= b, \\ \theta(a + c) &= c, \\ \theta(b + c) &= b + c. \end{aligned}$$

Since, we have for all $a \in A : \theta(a) = a$ we see at once that this operator is not linear. This operator is useful in protocol verifications. In [**40**] the priority operator has been used for the verification of the *PAR*-protocol. The acronym *PAR* stands for Positive Acknowledgement with Retransmission.

1.5. Conditional axioms

In this section we will discuss some additional axioms; the so-called conditional (alphabet) axioms. They are the subject of research in [**5**]. To formulate some of the conditions, the notion of the alphabet of a process is needed. We can think of this alphabet as the set of atomic actions that the process might perform. For example, the alphabet of $a+b$ will be the set containing a and b. It is known that the conditional axioms can be proved for closed terms, q.v. [**5**]. These axioms are an indispensable tool in algebraic verification techniques; cf. [**3**] and, of course, [**5**]. We will only use conditional axioms in chapter 2. The reason for mentioning them here is that we propose in chapter 3 an axiom system in which it will be possible to deduce some of the conditional axioms.

Definition (1.5.1) *Alphabet*

The alphabet $\alpha(x)$ of a process x is the set of atomic actions that can be executed by x, so $\alpha(x) \subseteq A$. The axioms are given in table 1.5. We use the following notational conventions: $a \in A$, $\gamma \in C$ and x and y are arbitrary processes.

$\alpha(\gamma) = \emptyset$	AB1
$\alpha(a) = \{a\}$	AB2
$\alpha(\tau \cdot x) = \alpha(x)$	AB3
$\alpha(a \cdot x) = \{a\} \cup \alpha(x)$	AB4
$\alpha(x+y) = \alpha(x) \cup \alpha(y)$	AB5

Table 1.5. Alphabet function.

$\alpha(x) = \bigcup_{n=1}^{\infty} \alpha\big(\pi_n(x)\big)$	AB6
$\alpha\big(\tau_I(x)\big) = \alpha(x) \setminus I$	AB7

Table 1.6. Extension to infinite processes.

The axioms in table 1.5 give us an inductive definition of $\alpha(x)$ on finite terms. Since all finite projections of processes that can be specified with the aid of a guarded recursive specification are closed terms [**8**], we can use axiom AB6 in table 1.6 to define the alphabet of this class of processes. We can extend this to yet a bigger class with the aid of AB7.

$\alpha(x) \mid \big(\alpha(y) \cap H\big) \subseteq H \Longrightarrow \partial_H(x \parallel y) = \partial_H\big(x \parallel \partial_H(y)\big)$	CA1
$\alpha(x) \mid \big(\alpha(y) \cap I\big) = \emptyset \Longrightarrow \tau_I(x \parallel y) = \tau_I\big(x \parallel \tau_I(y)\big)$	CA2
$\alpha(x) \cap H = \emptyset \Longrightarrow \partial_H(x) = x$	CA3
$\alpha(x) \cap I = \emptyset \Longrightarrow \tau_I(x) = x$	CA4
$H = H_1 \cup H_2 \Longrightarrow \partial_H(x) = \partial_{H_1} \circ \partial_{H_2}(x)$	CA5
$I = I_1 \cup I_2 \Longrightarrow \tau_I(x) = \tau_{I_1} \circ \tau_{I_2}(x)$	CA6
$H \cap I = \emptyset \Longrightarrow \tau_I \circ \partial_H(x) = \partial_H \circ \tau_I(x)$	CA7

Table 1.7. Conditional alphabet axioms.

The conditional alphabet axioms are defined in table 1.7. We will explain the "bar" notation in that table. Let $B, C \subseteq A$. Then $B \mid C$ is the set of all the non-trivial communications:

$$B \mid C = \{b \mid c : b \in B, c \in C\} \setminus \{\delta\}.$$

Chapter 2

On the Register Operator

THE main goal of this chapter is to introduce a formal notion for the parallel notation of a process with parallel input. We use a special state operator: the register operator. In this chapter it is our intention to stick as closely as possible to the theory ACP_τ and additional features. Nevertheless, it can be seen as a case study for the subsequent chapters. There, we will develop theories that will allow us to introduce operators like the register operator in a comfortable way. As an application of the theory that we will present in this chapter we will prove a result on systolic arrays.

2.1. Introduction

We will work within the axiomatic framework of ACP_τ and the register operator. The axioms of ACP_τ can be found in chapter 1. The register operator will be discussed in this chapter. It is an instance of the state operator such that it will distribute over the merge in the applications we are interested in.

In this section we will briefly discuss the structure of this chapter: in section 2.2 most of the definitions needed are stated. In section 2.3 we give three comprehensive examples in which we show how the register operator can be used to obtain compact (and correct) specifications of processes with parallel input. The second and third example deal with the "cells" of systolic arrays. The following question about the register operator immediately arose: "Do we have to eliminate the register operators in all the individual cells in order to be able to prove correctness theorems on systolic arrays?" In the sequel of this chapter we will answer this question in a negative way. Section 2.4 consists of the prerequisites we need, to prove the main theorem that is stated and proved in section 2.5. This theorem implies that the register operator distributes over the merge if the individual processes only use their own registers. In section 2.6 we will give, as an application of the theory, a correctness proof of a theorem on palindrome recognition. This theorem states that we can deal with sesquipedalian words if we simply merge sufficient palindrome recognizers that can handle only two-letter words.

2.2. Definitions

Let us consider a collection of datum elements D and a process P with parallel input of these datum elements. Let us add a datum element $\perp \in D$, denoting a meaningless default. We define a partial ordering on D as follows:

$$\text{if } x, y \in D \text{ then } x \leq y \iff x = y \text{ or } x = \perp.$$

Let $t \in \mathbb{N}$ and let R be the following collection:

$$R = \{\, (d_1, \ldots, d_t) : d_i \in D, \text{ and } 1 \leq i \leq t,\ t \in \mathbb{N} \,\}.$$

Let $\{f_1, \ldots, f_k\}$ be the collection of functions of the following form that occur in the data structure of the atomic actions of P:

$$f_i : \underbrace{D \times D \times \cdots \times D}_{k_i \text{ times}} \longrightarrow D.$$

Observe that $\{f_1, \ldots, f_k\}$ is never empty since $id : D \longrightarrow D$ is always in $\{f_1, \ldots, f_k\}$, that is, if we have an atomic action $s(d)$ then the function occurring in it can be seen as id: $s\big(id(d)\big)$. Let $[D^n \longrightarrow D]$ denote the collection of all the functions with D^n as domain and D as range. We will give hereinafter an inductive definition of a subset F of $\bigcup_{n=1}^{t} [D^n \longrightarrow D]$.

(*i*) $f_1, \ldots, f_k \in F$

(*ii*) $\wedge \in F$

where $\wedge$ is: if $x, y \in D :\ x \wedge y = \inf\{x, y\}$

(*iii*) $\forall\, 1 \leq k \leq t\, \forall\, 1 \leq i \leq k :\quad p_i^k : D^k \longrightarrow D \in F,$

where p_i^k is: if $(e_1, \ldots, e_k) = \vec{e} \in D^k$, then $p_i^k(\vec{e}) = e_i$

(*iv*) $\forall\, d \in D\ \forall\, 1 \leq k \leq t :\quad \hat{d}^k : D^k \longrightarrow D \in F,$

where $\hat{d}^k$ is: $\forall \vec{e} \in D^k : \hat{d}^k(\vec{e}) = d$

(*v*) $f : D^n \longrightarrow D,\ g_1, \ldots, g_n : D^l \longrightarrow D \in F$

$\Longrightarrow f(g_1, \ldots, g_n) : D^l \longrightarrow D \in F$

where $f(g_1, \ldots, g_n)$ is defined as follows:

if $(e_1, \ldots, e_l) = \vec{e} \in D^l$, then $f(g_1, \ldots, g_n) = f\big(g_1(\vec{e}), \ldots, g_n(\vec{e})\big)$.

In fact the notation of the constant functions and the projections is a little unfathomable; so for the constant functions we prefer to use the constants themselves, and for the projections we leave out the arity.

Let U be the following finite set:

$$U = \{\, u_1, \ldots, u_n \,\} \cup \{\, \Delta \,\}.$$

Choose a function $|: U^2 \longrightarrow U$ which is commutative, associative, and defined everywhere. Thus for all $u, v, w \in U$:

$$\begin{aligned} u \mid v &\in U \\ u \mid \Delta &= \Delta \\ u \mid v &= v \mid u \\ (u \mid v) \mid w &= u \mid (v \mid w) \end{aligned}$$

With this function we will define the communication function on the atomic actions as follows. For all $d, d' \in D$ and for all $u, v \in U$ we have:

$$u(d) \mid v(d') = (u \mid v)(d \wedge d').$$

In which we define for all $u \in U: \; u(\perp) = \delta$. Furthermore we assume for all $d \in D$ that $\Delta(d) = \delta$. As an illustration consider the following. Suppose for the moment that we deal with commonly used communication in *ACP*, then we have say $U = \{r, s, c, \Delta\}$ and $\mid$ is defined as follows:

$$r \mid s = s \mid r = c,$$

and all the other combinations result in Δ. We will now verify that we have read/send communication. For,

$$\begin{aligned} r(d) \mid s(d) &= (r \mid s)(d \wedge d) \\ &= c(d \wedge d) \\ &= c(d) \end{aligned}$$

All the other communications result in δ.

Let $\mathbb{N}_t = \{1, 2, \ldots, t\}$ and $r = (d_1, \ldots, d_t) \in R$. Define a function

$$\uparrow \; : R \times \mathbb{N}_t \longrightarrow D$$

by

$$r \uparrow n = d_n.$$

Let $f \in F$ be a p-ary function and define a function

$$f(\uparrow n_1, \ldots, \uparrow n_p) : R \longrightarrow D$$

by

$$r \mapsto f(r \uparrow n_1, \ldots, r \uparrow n_p), \quad \text{for } n_1, \ldots, n_p \in \mathbb{N}_t.$$

Apparently there is also an output of P in which the parallel input of P is being used in some way. In other words we want to store the input of P. For this purpose we can define some new atomic actions. We will also need some new atomic actions for reading the input, when we need that input. Below we will list all the new atomic actions. First some notational machinery: a row of elements $n_1, \ldots, n_k$ is represented by N_k, the row $\uparrow n_1, \ldots, \uparrow n_k$ is denoted by $\uparrow N_k$, the row $r \uparrow n_1, \ldots, r \uparrow n_k$ by $r \uparrow N_k$, and, provided that no confusion arises, for $\{n_1, \ldots, n_k\}$ we simply write N_k instead of $\{N_k\}$.

We define $U(D) = \{\, u(d) \mid u \in U,\, d \in D \,\}$. Now let $u(d) \in U(D)$, then for all rows N_k needed, we define a new atom: $u(d \downarrow N_k)$. The intuition behind this new action is as follows: $r(d \downarrow 1, 2)$ means that we read a d and afterwards we want to store this d on the first and second position. We can do this with the register operator. The collection of this type of atomic actions is denoted by $U(D \downarrow)$.

Let $f \in F$ and $u \in U$, then for all the necessary rows N_k we define a new atom: $u\big(f(\uparrow N_k)\big)$ and we denote the collection of all these atomic actions by $U(F)$. As an exemplification of this new atomic action, consider the following: $s\big(id(\uparrow 1)\big)$. We want to sent a datum element, but in this phase we do not know which one. With the aid of the register operator we can read what datum element should be sent. In case we want to store the output directly we have for all the necessary rows M_l an atom $u\big(f(\uparrow N_k) \downarrow M_l\big)$ and the collection of all these atomic actions is denoted by $U(F \downarrow)$. The meaning of this type of atoms is the same as the former type. The only difference is that we can store the datum in a possibly other register. After the actual definition of the register operator we will illustrate how the last mentioned category of atomic actions may be used, see example (2.2.1).

We have now made a collection of atomic actions

$$A = A' \cup U(D) \cup U(D \downarrow) \cup U(F) \cup U(F \downarrow),$$

in which A' is the collection of atomic actions that are not in

$$U(D) \cup U(D \downarrow) \cup U(F) \cup U(F \downarrow).$$

We will extend the communication function $\mid$ to this new set of atomic actions A. Suppose that we have $u \mid v = w$ for some $u, v, w \in U$. Henceforward we use instead of $(u \mid v)(d \wedge d)$ the formula $w(d)$:

$$
\begin{aligned}
&u(d) \mid v(d \downarrow N_k) = w(d \downarrow N_k)\\
&u(d) \mid v\big(f(\uparrow N_k)\big) = w\big((d \wedge f)(\uparrow N_k)\big)\\
&u(d) \mid v\big(f(\uparrow N_k) \downarrow M_l\big) = w\big((d \wedge f)(\uparrow N_k) \downarrow M_l\big)\\
&u(d \downarrow N_k) \mid v(d \downarrow M_l) = w(d \downarrow N_k, M_l)\\
&u(d \downarrow N_k) \mid v\big(f(\uparrow M_l)\big)\\
&\qquad = \begin{cases} w\big((d \wedge f)(\uparrow M_l) \downarrow N_k\big), & \text{if } N_k \cap M_l = \emptyset;\\ \delta, & \text{otherwise.}\end{cases}
\end{aligned}
$$

$$u(d \downarrow N_k) \mid v\big(f(\uparrow M_l) \downarrow P_n\big)$$
$$= \begin{cases} w\big((d \wedge f)(\uparrow M_l) \downarrow N_k, P_n\big), & \text{if } N_k \cap M_l = \emptyset; \\ \delta, & \text{otherwise.} \end{cases}$$
$$u\big(f(\uparrow N_k)\big) \mid v\big(g(\uparrow M_l)\big) = w\big((f' \wedge g')(\uparrow S_u)\big) \qquad (N_k \cup M_l = S_u)$$
$$u\big(f(\uparrow N_k)\big) \mid v\big(g(\uparrow M_l) \downarrow P_n\big)$$
$$= \begin{cases} w\big((f' \wedge g')(\uparrow S_u) \downarrow P_n\big), & \text{if } N_k \cap P_n = \emptyset; \\ \delta, & \text{otherwise.} \end{cases}$$
$$u\big(f(\uparrow N_k) \downarrow P_n\big) \mid v\big(g(\uparrow M_l) \downarrow Q_s\big)$$
$$= \begin{cases} w\big((f' \wedge g')(\uparrow S_u) \downarrow P_n, Q_s\big), & \text{if } N_k \cap Q_s = M_l \cap P_n = \emptyset; \\ \delta, & \text{otherwise.} \end{cases}$$

Now we will elucidate the "aggrandizement" of the communication function, and—first of all—the meaning of $d \wedge f$ and $f' \wedge g'$.

$$d \wedge f = \hat{d}^k \wedge f : D^k \longrightarrow D$$

and

$$f' \wedge g' = f(p_{n_1}, \ldots, p_{n_k}) \wedge g(p_{m_1}, \ldots, p_{m_l}) : D^u \longrightarrow D$$

where

$$\big|N_k \cup M_l\big| = u,\ N_k \cup M_l = S_u,\ \text{and } p_{s_1}, \ldots, p_{s_u} : D^u \longrightarrow D.$$

The first communication is simple, e.g., if we have $r(d) \mid s(d \downarrow 1)$, we want to be able to store d in a register on the first position after the communication has taken place. Thence the resulting communication is $c(d \downarrow 1)$. In the second formula we use "generic" communication. As an example we take

$$r(d) \mid s\big(id(\uparrow 1)\big) = c\big(d \wedge id(\uparrow 1)\big).$$

If this communication takes place, we do not know what value the register operator will contain on the first position. If it happens to be a d, we really want this communication. If not, then $c\big(d \wedge id(\uparrow 1)\big)$ should be δ. In the third formula the only difference with the second is that, if we indeed have a d, we want to store it in a possibly other register. The fourth formula is easy to understand: if we have $r(d \downarrow 1) \mid s(d \downarrow 2)$, we want to store the d on both positions in the register. The fifth formula yields almost the same result as the third, albeit we have made a restriction as to the registers that can be used. For, if we should have $r(d \downarrow 1) \mid s\big(id(\uparrow 1)\big)$, then we have a "timing" problem: if the d is stored at position 1 in the register before it is read, we obtain, in general, another answer, than if we first read the register and then store the d at position 1. We do not have this problem in the third formula. For first we must read what we have in the registers, and then we store the result. In all the other formulas we state this sort of conditions for the same reason (we will call them the register conditions for later reference). The sixth formula is

essentially the same as the fifth. In all the other formulas we have two atomic actions that might communicate. And they do if it turns out that f and g coincide if they are instantiated by the values of the current register, i.e., if $f(d_{n_1}, \dots, d_{n_k}) = g(d_{m_1}, \dots, d_{m_l})$. Below we will axiomatize what the register operator is, and that the "past tense" of all these new atoms will be the old atomic actions that we already had.
Now choose an $r = (d_1, \dots, d_t) \in R$. For such an r we will introduce a new operator that we denote by $[r]$. With $[r(N_k/d)]$ we mean $[r]$, but on the n_i-th place a d instead of d_{n_i}, $(1 \leq i \leq k)$. Below we will give a complete list of axioms:

$$[r](\gamma) = \gamma$$
$$[r](\gamma \cdot x) = \gamma \cdot [r](x)$$
$$[r]\big(u(d \downarrow N_k)\big) = u(d)$$
$$[r]\big(u\big(f(\uparrow N_k)\big)\big) = u\big(f(r \uparrow N_k)\big)$$
$$[r]\big(u\big(f(\uparrow N_k) \downarrow M_l\big)\big) = u\big(f(r \uparrow N_k)\big)$$
$$[r](u(d \downarrow N_k) \cdot x) = u(d) \cdot [r(N_k/d)](x)$$
$$[r]\big(u\big(f(\uparrow N_k)\big) \cdot x\big) = u\big(f(r \uparrow N_k)\big) \cdot [r](x)$$
$$[r]\big(u\big(f(\uparrow N_k) \downarrow M_l\big) \cdot x\big) = u\big(f(r \uparrow N_k)\big) \cdot [r\,(M_l/f(r \uparrow N_k))]\,(x)$$

For all other atomic actions we have:

$$[r](a) = a$$
$$[r](a \cdot x) = a \cdot [r](x)$$

And finally:

$$[r](x + y) = [r](x) + [r](y).$$

Where γ is a special constant, e.g., δ or τ, x and y are arbitrary processes.

Example (2.2.1)

Consider the following guarded recursive specification:

$$E = \big\{\, Y = Y(0),\ Y(n) = s(n+1) \cdot Y(n+1) \;\big|\; n \in \mathbb{N} \,\big\}.$$

It will be clear that the following process is a solution for E:

$$\prod_{n=1}^{\infty} s(n) = s(1) \cdot s(2) \cdot s(3) \cdot \cdots .$$

Let f be the successor function on $\mathbb{N}$. Define $X' = s\big(f(\uparrow 1) \downarrow 1\big) \cdot X'$. Let $X = [0](X')$. We obviously have: $X = Y$. For, if we put

$$\forall n \in \mathbb{N}: \quad Y(n) := [n](X'),$$

then this system is also a solution for E. Thence, by RSP, we obtain

$$X = Y = \prod_{n=1}^{\infty} s(n).$$

Observe that we introduced here infinitely many atomic actions. If we should discuss this topic in detail, it would be better to define f on $\{0, 1, \ldots, k-1\}$ with $f(k-1) = 0$. In that case we obtained:

$$X = \left(\prod_{i=0}^{k-1} s(i)\right)^{\omega}.$$

Remark (2.2.2)

In fact, the register operator is an instance of the state operator. For the definition of the state operator we refer to chapter 1, but for a more concise treatment we refer to [**4**]. The main difference here is that we denoted it by $[r]$ instead of λ_r. This information is redundant in order to understand the theory presented in this chapter. However, for reasons of completeness we will enumerate here the action function and the effect function of this state operator.

Since there is only one object, we take $M = \{m\}$. For the state space S we take the set R.

The action function:
$u(d \downarrow N_k)(m, r) = u(d)$,
$u\big(f(\uparrow N_k)\big)(m, r) = u\big(f(r \uparrow N_k)\big)$,
$u\big(f(\uparrow N_k) \downarrow M_l\big)(m, r) = u\big(f(r \uparrow N_k)\big)$.
And for the other atomic actions $a \in A$ we have:
$a(m, r) = a$.

The effect function:
$r(m, u(d \downarrow N_k)) = r(N_k/d)$,
$r\big(m, u\big(f(\uparrow N_k) \downarrow M_l\big)\big) = r\big(M_l/f(r \uparrow N_k)\big)$.
And otherwise:
$r(m, a) = r$.

2.3. Hermeneutics

In this section we will explain the symbolism of the previous section by giving three examples. Consider the following process P: first we want to read data via channels 1 and 2 in an arbitrary order (parallel input). Afterwards we want to send the sum of these datum elements—which itself is a datum element—via channel 3. Then the process starts all over again. In [**44**] we see that such processes are specified as follows:

$$P = \Big(\sum_{d\in D} r_1(d) \parallel \sum_{e\in D} r_2(e)\Big) \cdot s_3(d+e) \cdot P, \tag{1}$$

although it is a well-known fact that this way of specifying is at least questionable, for, the d in the sum is a bound variable. Hence we can rename this d without changing the semantics of the formula. But the d in $s_3(d+e)$ is not in the scope of the first sum.

Since we often have an interleaving merge this dubious way of specifying the process P is said to be shorthand for

$$\begin{aligned} P = {}& \sum_{d\in D} r_1(d) \cdot \sum_{e\in D} r_2(e) \cdot s_3(d+e) \cdot P \\ & + \sum_{e\in D} r_2(e) \cdot \sum_{d\in D} r_1(d) \cdot s_3(d+e) \cdot P. \end{aligned} \tag{2}$$

Of course, we do not want to write down equation (2) if we mean equation (1) all the time. We see that in equation (2) there are no parentheses to show where the scopes of the sum signs end. Note that P is a process, so in P there are no variables. This means that we can freely choose the position of these delimiters:

$$\begin{aligned} P &= \sum_{d\in D}\Big(r_1(d) \cdot \sum_{e\in D}\big(r_2(e) \cdot s_3(d+e)\big)\Big) \cdot P \\ &\quad + \sum_{e\in D}\Big(r_2(e) \cdot \sum_{d\in D}\big((r_1(d) \cdot s_3(d+e)\big)\Big) \cdot P \\ &= \sum_{d\in D}\Big(r_1(d) \cdot \sum_{e\in D}\big(r_2(e) \cdot s_3(d+e) \cdot P\big)\Big) \\ &\quad + \sum_{e\in D}\Big(r_2(e) \cdot \sum_{d\in D}\big(r_1(d) \cdot s_3(d+e) \cdot P\big)\Big). \end{aligned} \tag{3}$$

This is an immediate consequence of the left distributivity of the sequential composition.

We will show how we can adapt equation (1) such that the scope problem is solved. Let $f : D \times D \longrightarrow D$ be the function that represents the sum of two datum elements and assume for the set of atoms A the following:

$$\begin{aligned} &\{\, r_i(d \downarrow i), c_i(d \downarrow i) \mid d \in D,\ i = 1, 2 \,\} \\ &\qquad \cup \{\, s_3\big(f(\uparrow 1, \uparrow 2)\big), c_3\big((d \wedge f)(\uparrow 1, \uparrow 2)\big) \mid d \in D \,\} \subseteq A. \end{aligned}$$

Consider the following specification:

$$\begin{aligned} X &= \Big(\sum_{d\in D} r_1(d\downarrow 1) \parallel \sum_{e\in D} r_2(e\downarrow 2)\Big)\cdot Y \\ Y &= s_3\big(f(\uparrow 1, \uparrow 2)\big)\cdot X \end{aligned}$$

Theorem (2.3.1)

$\forall\, [r]\in R : [r](X) = P.$

Proof. Let $[r] = [u,v] \in R$ be chosen arbitrarily. Since we have interleaving we immediately see:

$$X = \sum_{d\in D} r_1(d\downarrow 1)\cdot \sum_{e\in D} r_2(e\downarrow 2)\cdot Y + \sum_{e\in D} r_2(e\downarrow 2)\cdot \sum_{d\in D} r_1(d\downarrow 1)\cdot Y.$$

Hence we get

$$\begin{aligned} [r](X) &= \sum_{d\in D} r_1(d)\cdot [d,v]\Big(\sum_{e\in D} r_2(e\downarrow 2)\cdot Y\Big) \\ &\quad + \sum_{e\in D} r_2(e)[u,e]\Big(\sum_{d\in D} r_1(d\downarrow 1)\cdot Y\Big) \\ &= \sum_{d\in D} r_1(d)\cdot \sum_{e\in D} r_2(e)[d,e](Y) \\ &\quad + \sum_{e\in D} r_2(e)\cdot \sum_{d\in D} r_1(d)[d,e](Y). \end{aligned} \tag{4}$$

Now we are going to calculate $[d,e](Y)$:

$$\begin{aligned} [d,e](Y) &= [d,e]\big(s_3\big(f(\uparrow 1,\uparrow 2)\big)\cdot X\big) \\ &= s_3\big(f(d,e)\big)\cdot [d,e](X) \\ &= s_3(d+e)\cdot [d,e](X). \end{aligned} \tag{5}$$

Combining the equations (4) and (5) we find thus:

$$\begin{aligned} [r](X) &= \sum_{d\in D} r_1(d)\cdot \sum_{e\in D} r_2(e)\cdot s_3(d+e)\cdot [d,e](X) \\ &\quad + \sum_{e\in D} r_2(e)\cdot \sum_{d\in D} r_1(d)\cdot s_3(d+e)\cdot [d,e](X). \end{aligned} \tag{6}$$

It is evident that $[d,e](X)$ in the first term of (6) differs, in general, from $[d,e](X)$ in the second term of (6). To avoid possible ambiguity we will rename the bound variables d and e in the second part of the right-hand side of equation (6), and so we obtain:

$$\begin{aligned} [r](X) &= \sum_{d\in D} r_1(d)\cdot \sum_{e\in D} r_2(e)\cdot s_3(d+e)\cdot [d,e](X) \\ &\quad + \sum_{g\in D} r_2(g)\cdot \sum_{f\in D} r_1(f)\cdot s_3(f+g)\cdot [f,g](X). \end{aligned} \tag{7}$$

Let us now index the set R by saying $R = \{ r_j \mid 1 \le j \le |D|^2 \}$. Define a function

$$i : D \times D \longrightarrow \mathbb{N}_{|D|^2} \quad \text{by } i(d,e) = j \iff r_j = [d,e].$$

Consider the guarded recursive specification E below:

$$E = \Big\{ X_i = \sum_{d \in D} \Big(r_1(d) \cdot \sum_{e \in D} \big(r_2(e) \cdot s_3(d+e) \cdot X_{i(d,e)} \big) \Big) + \sum_{g \in D} \Big(r_2(g) \cdot \sum_{f \in D} \big(r_1(f) \cdot s_3(f+g) \cdot X_{i(f,g)} \big) \Big) \;\Big|\; 1 \le i \le |D|^2 \Big\}.$$

With the aid of the equalities in equation (3) we immediately see that, if we put

$$P = X_i = X_{i(d,e)} = X_{i(f,g)} \quad \text{for all } i \in \mathbb{N}_{|D|^2},$$

that P is a solution for E. However, if we put

$$X_i = [r_i](X),\ X_{i(d,e)} = [d,e](X),\ \text{and } X_{i(f,g)} = [f,g](X) \text{ for all } i \in \mathbb{N}_{|D|^2},$$

we find, because of equation (7), that this system is also a solution for E. Hence, we can conclude now, with the aid of *RSP*, that $[r](X) = P$ for all $r \in R$. This proves 2.3.1.

Now we will consider another example that is known as an *IPS* cell. See, e.g., [**33**]. For the sake of convenience we have somewhat simplified notations. The specification of P' at present looks like this:

$$\begin{aligned} P' &= \Big(\sum_{a \in D} r_1(a) \parallel \sum_{b \in D} r_2(b) \parallel \sum_{x \in D} r_3(x) \Big) \cdot Q'(a,b,x) \\ Q'(a,b,x) &= \big(s_3(b) \parallel s_2(x + a \cdot b) \big) \cdot P'. \end{aligned} \tag{8}$$

Now we are going to eliminate the merge in the first equation of (8) in order to achieve a correct specification of P':

$$\begin{aligned} P' = \sum_{a \in D} r_1(a) \cdot \Big(&\sum_{b \in D} r_2(b) \cdot \sum_{x \in D} r_3(x) \cdot Q'(a,b,x) \\ &+ \sum_{x \in D} r_3(x) \cdot \sum_{b \in D} r_2(b) \cdot Q'(a,b,x) \Big) \\ &+ \sum_{b \in D} r_2(b) \cdot \Big(\sum_{a \in D} r_1(a) \cdot \sum_{x \in D} r_3(x) \cdot Q'(a,b,x) \\ &+ \sum_{x \in D} r_3(x) \cdot \sum_{a \in D} r_1(a) \cdot Q'(a,b,x) \Big) \\ &+ \sum_{x \in D} r_3(x) \cdot \Big(\sum_{a \in D} r_1(a) \cdot \sum_{b \in D} r_2(b) \cdot Q'(a,b,x) \\ &+ \sum_{b \in D} r_2(b) \cdot \sum_{a \in D} r_1(a) \cdot Q'(a,b,x) \Big), \end{aligned}$$

$$Q'(a,b,x) = \big(s_3(b) \parallel s_2(x + a \cdot b) \big) \cdot P'. \tag{9}$$

Let the function f be defined as follows:

$$f : D \times D \times D \longrightarrow D \quad \text{with } f(a,b,x) = x + a \cdot b, \text{ for all } a,b,x \in D.$$

In this example we also need the function $id : D \longrightarrow D$. Suppose now that the following holds for the set A of atomic actions:

$$\{\, r_i(d \downarrow i) \mid d \in D,\ i = 1,2,3 \,\} \cup \{ s_3\big(id(\uparrow 2)\big),\ s_2\big(f(\uparrow 1, \uparrow 2, \uparrow 3)\big) \} \subseteq A.$$

Observe that we only listed the atomic actions needed for the specification below. In fact we also introduced, by adding the aforementioned atoms, a number of communication actions. Neither are these atoms necessary to give a correct specification, nor do they occur in the correctness proof hereinafter. Nevertheless we will now give an enumeration of all new atomic actions:

$$\begin{aligned}
&\{\, r_i(d \downarrow i), c_i(d \downarrow i) \mid d \in D,\ i = 1,2,3 \,\}\\
&\quad \cup \{\, c_3\big((d \wedge id)(\uparrow 2)\big), c_3\big((d \wedge id)(\uparrow 2) \downarrow 1\big) \mid d \in D \,\}\\
&\quad \cup \{\, c_2\big((d \wedge f)(\uparrow 1, \uparrow 2, \uparrow 3)\big), c_2\big((d \wedge f)(\uparrow 1, \uparrow 2, \uparrow 3) \downarrow 2\big) \mid d \in D \,\}\\
&\quad \cup \{ s_3\big(id(\uparrow 2)\big),\ s_2\big(f(\uparrow 1, \uparrow 2, \uparrow 3)\big) \}.
\end{aligned}$$

With these new atomic actions in view we can write down the following equations:

$$X' = \Big(\sum_{a \in D} r_1(a \downarrow 1) \parallel \sum_{b \in D} r_2(b \downarrow 2) \parallel \sum_{x \in D} r_3(x \downarrow 3) \Big) \cdot Y', \tag{10}$$

$$Y' = \Big(s_3\big(id(\uparrow 2)\big) \parallel s_2\big(f(\uparrow 1, \uparrow 2, \uparrow 3)\big) \Big) \cdot X'.$$

Theorem (2.3.2)

$\forall\, [r] \in R : [r](X') = P'.$

Proof. Let $[r] = [u,v,w] \in R$ be chosen arbitrarily. Below we have eliminated the merge in equation (10):

$$\begin{aligned}
X' = \sum_{a \in D} r_1(a \downarrow 1) \cdot \Big(&\sum_{b \in D} r_2(b \downarrow 2) \cdot \sum_{x \in D} r_3(x \downarrow 3) \cdot Y'\\
&+ \sum_{x \in D} r_3(x \downarrow 3) \cdot \sum_{b \in D} r_2(b \downarrow 2) \cdot Y' \Big)\\
+ \sum_{b \in D} r_2(b \downarrow 2) \cdot \Big(&\sum_{a \in D} r_1(a \downarrow 1) \cdot \sum_{x \in D} r_3(x \downarrow 3) \cdot Y'\\
&+ \sum_{x \in D} r_3(x \downarrow 3) \cdot \sum_{a \in D} r_1(a \downarrow 1) \cdot Y' \Big)\\
+ \sum_{x \in D} r_3(x \downarrow 3) \cdot \Big(&\sum_{a \in D} r_1(a \downarrow 1) \cdot \sum_{b \in D} r_2(b \downarrow 2) \cdot Y'\\
&+ \sum_{b \in D} r_2(b \downarrow 2) \cdot \sum_{a \in D} r_1(a \downarrow 1) \cdot Y' \Big).
\end{aligned}$$

Hence if we apply $[r]$ we obtain:

$$\begin{aligned}
[r](X') = \sum_{a\in D} r_1(a) \cdot \Big(&\sum_{b\in D} r_2(b) \cdot \sum_{x\in D} r_3(x) \cdot [a,b,x](Y') \\
&+ \sum_{x\in D} r_3(x) \cdot \sum_{b\in D} r_2(b) \cdot [a,b,x](Y')\Big) \\
&+ \sum_{b\in D} r_2(b) \cdot \Big(\sum_{a\in D} r_1(a) \cdot \sum_{x\in D} r_3(x) \cdot [a,b,x](Y') \\
&+ \sum_{x\in D} r_3(x) \cdot \sum_{a\in D} r_1(a) \cdot [a,b,x](Y')\Big) \\
&+ \sum_{x\in D} r_3(x) \cdot \Big(\sum_{a\in D} r_1(a) \cdot \sum_{b\in D} r_2(b) \cdot [a,b,x](Y') \\
&+ \sum_{b\in D} r_2(b) \cdot \sum_{a\in D} r_1(a) \cdot [a,b,x](Y')\Big)
\end{aligned}$$

Let us now calculate $[a,b,x](Y')$:

$$\begin{aligned}
[a,b,x](Y') &= [a,b,x]\Big(s_3\big(id(\uparrow 2)\big) \cdot s_2\big(f(\uparrow 1, \uparrow 2, \uparrow 3)\big) \cdot X' \\
&\qquad + s_2\big(f(\uparrow 1, \uparrow 2, \uparrow 3)\big) \cdot s_3\big(id(\uparrow 2)\big) \cdot X'\Big) \\
&= s_3(b) \cdot [a,b,x]\big(f(\uparrow 1, \uparrow 2, \uparrow 3)\big) \cdot X'\big) \\
&\qquad + s_2(x + a \cdot b) \cdot [a,b,x]\big(s_3\big(id(\uparrow 2)\big) \cdot X'\big) \\
&= s_3(b) \cdot s_2(x + a \cdot b) \cdot [a,b,x](X') \\
&\qquad + s_2(x + a \cdot b) \cdot s_3(b) \cdot [a,b,x](X') \\
&= \big(s_3(b) \parallel s_2(x + a \cdot b)\big) \cdot [a,b,x](X'). && (11)
\end{aligned}$$

Notice that we actually have computed one of the six possibilities. The others are calculated analogously. However, to avoid ambiguity we will rename some of the bound variables that occur in $[r](X')$. This results in:

$$\begin{aligned}
[r](X') = \sum_{a\in D} r_1(a) \cdot \Big(&\sum_{b\in D} r_2(b) \cdot \sum_{x\in D} r_3(x) \cdot [a,b,x](Y') \\
&+ \sum_{y\in D} r_3(y) \cdot \sum_{c\in D} r_2(c) \cdot [a,c,y](Y')\Big) \\
&+ \sum_{e\in D} r_2(e) \cdot \Big(\sum_{d\in D} r_1(d) \cdot \sum_{z\in D} r_3(z) \cdot [d,e,z](Y') \\
&+ \sum_{t\in D} r_3(t) \cdot \sum_{f\in D} r_1(f) \cdot [f,e,t](Y')\Big) \\
&+ \sum_{s\in D} r_3(s) \cdot \Big(\sum_{g\in D} r_1(g) \cdot \sum_{h\in D} r_2(h) \cdot [g,h,s](Y')
\end{aligned}$$

$$+\sum_{m\in D} r_2(m)\cdot\sum_{l\in D} r_1(l)\cdot[l,m,s](Y')\Big). \tag{12}$$

Let $\{\, r_j \mid 1 \leq j \leq |D|^3 \,\}$ be an indexation of the set R, and define:

$$i : D\times D\times D \longrightarrow \mathbb{N}_{|D|^3}$$

with, for all $d, e, f \in D$,

$$i(d,e,f) = j \iff r_j = [d,e,f],$$

we can write down the following guarded recursive specification:

$$\begin{aligned}
E' = \Big\{\, X'_i = {} & \sum_{a\in D} r_1(a)\cdot\Big(\sum_{b\in D} r_2(b)\cdot\sum_{x\in D} r_3(x)\\
&\cdot\big(s_3(b)\parallel s_2(x+a\cdot b)\big)\cdot X'_{i(a,b,x)}\\
&+\sum_{y\in D} r_3(y)\cdot\sum_{c\in D} r_2(c)\cdot\big(s_3(c)\parallel s_2(y+a\cdot c)\big)\cdot X'_{i(a,c,y)}\Big)\\
&+\sum_{e\in D} r_2(e)\cdot\Big(\sum_{d\in D} r_1(d)\cdot\sum_{z\in D} r_3(z)\cdot\big(s_3(e)\parallel s_2(z+d\cdot e)\big)\cdot X'_{i(d,e,z)}\\
&+\sum_{t\in D} r_3(t)\cdot\sum_{f\in D} r_1(f)\cdot\big(s_3(e)\parallel s_2(t+f\cdot e)\big)\cdot X'_{i(f,e,t)}\Big)\\
&+\sum_{s\in D} r_3(s)\cdot\Big(\sum_{g\in D} r_1(g)\cdot\sum_{h\in D} r_2(h)\cdot\big(s_3(h)\parallel s_2(s+g\cdot h)\big)\cdot X'_{i(g,h,s)}\\
&+\sum_{m\in D} r_2(m)\cdot\sum_{l\in D} r_1(l)\\
&\cdot\big(s_3(m)\parallel s_2(s+l\cdot m)\big)\cdot X'_{i(l,m,s)}\Big) \;\Big|\; 1\leq i\leq |D|^3 \,\Big\}.
\end{aligned}$$

It is obvious that, if we put for all $i \in \mathbb{N}_{|D|^3}$,

$$P' = X'_i = X'_{i(a,b,x)} = X'_{i(a,c,y)} = X'_{i(d,e,z)} = X'_{i(f,e,t)} = X'_{ig,h,s} = X'_{i(l,m,s)}$$

that P' is a solution for E', because of the equations in (9). On the other hand, however, if we put for all $\mathbb{N}_{|D|^3}$:

$$\begin{aligned}
X'_i &= [r_i](X')\\
X'_{i(a,b,x)} &= [a,b,x](X')\\
X'_{i(a,c,y)} &= [a,c,y](X')\\
X'_{i(d,e,z)} &= [d,e,z](X')
\end{aligned}
\qquad\text{and}\qquad
\begin{aligned}
X'_{i(f,e,t)} &= [f,e,t](X')\\
X'_{i(g,h,s)} &= [g,h,s](X')\;,\\
X'_{i(l,m,s)} &= [l,m,s](X')
\end{aligned}$$

we immediately see, with (11) and (12), that this system is also a solution for the guarded recursive specification E'. Hence, with the use of *RSP* this concludes the proof of 2.3.2.

We are now about to give a last example, which is not very difficult, but we will need this example later on. In [**44**] we find in table 24 on page 135 a specification in which occurs a process $C_i'(x)$. Below we see this specification after we rid ourselves of all the superfluous notational ballast.

$$P'' = P''(x) = \Big(\sum_{y\in S} r_2(y) \parallel \sum_{v\in B} r_1(v)\Big)\cdot Q''(x,y,v) \tag{13}$$
$$Q''(x,y,v) = \big(s_2\big((x=y)\ \&\ v\big) \parallel s_1(y)\big)\cdot P''.$$

We will eliminate the merge in equation (13) in order to achieve a correct specification. Thus we obtain:

$$\begin{aligned} P'' = {} & \sum_{y\in S} r_2(y)\cdot \sum_{v\in B} r_1(v)\cdot Q''(x,y,v) \\ & + \sum_{v\in B} r_1(v)\cdot \sum_{y\in S} r_2(y)\cdot Q''(x,y,v) \end{aligned} \tag{14}$$
$$Q''(x,y,v) = \big(s_2\big((x=y)\ \&\ v\big) \parallel s_1(y)\big)\cdot P''. \tag{15}$$

Clearly we have $D := S\cup B$, with $B := \{\, \mathit{true},\ \mathit{false}\}$. Let us now define a function $f_x : D\times D \longrightarrow D$ by:

$$f_x(b,y) = \begin{cases} \mathit{true}, & \text{if } y\in S,\ b\in B,\ b=\mathit{true}\ \&\ (x=y); \\ \mathit{false}, & \text{otherwise.} \end{cases} \tag{16}$$

About the collection A of atomic actions we assume:

$$\{\, r_i(d\downarrow i) \mid d\in D,\ i=1,2\,\} \cup \{\, s_2\big(f_x(\uparrow 1,\uparrow 2)\big), s_1\big(id(\uparrow 2)\big)\,\} \subseteq A.$$

As before we only denoted the actions that we need in order to obtain the specification below. However, in the correctness proof on palindrome recognition in section 2.6 (theorem (2.6.2)), we will see a communication between a "new" atom and an "old" one. Thence, we will give all the new atomic actions in a separate display:

$$\begin{aligned} &\{\, r_i(d\downarrow i), c_i(d\downarrow i) \;\big|\; d\in D, i=1,2\,\} \\ &\quad\cup \{\, c_2\big((d\wedge f_x)(\uparrow 1,\uparrow 2)\big), c_2\big((d\wedge f_x)(\uparrow 1,\uparrow 2)\downarrow 2\big) \;\big|\; d\in D\,\} \\ &\quad\cup \{\, c_1\big((d\wedge id)(\uparrow 2)\big), c_1\big((d\wedge id)(\uparrow 2)\downarrow 1\big) \;\big|\; d\in D\,\} \\ &\quad\cup \{\, s_2\big(f_x(\uparrow 1,\uparrow 2)\big), s_1\big(id(\uparrow 2)\big)\,\}. \end{aligned}$$

We are now in a position to write down the following specification:

$$X'' = \Big(\sum_{y\in S} r_2(y\downarrow 2) \parallel \sum_{v\in B} r_1(v\downarrow 1)\Big)\cdot Y'' \tag{17}$$
$$Y'' = \big(s_2\big(f_x(\uparrow 1,\uparrow 2)\big) \parallel s_1\big(id(\uparrow 2)\big)\big)\cdot X'' \tag{18}$$

Theorem (2.3.3)

$\forall\, [r] \in R: \quad ACP \;\vdash\; [r](X'') = P''.$

Proof. Choose an $[r] = [d, e] \in R$. If we eliminate the merge in equation (17) and apply $[r]$ to it we immediately see that:

$$[r](X'') = \sum_{y \in S} r_2(y) \cdot \sum_{v \in B} r_1(v) \cdot [v, y](Y'') + \sum_{v \in B} r_1(v) \cdot \sum_{y \in S} r_2(y) \cdot [v, y](Y''). \quad (19)$$

It is also easy to see that:

$$[v, y](Y'') = \big(s_2\big((x = y) \;\&\; v\big) \parallel s_1(y)\big) \cdot [v, y](X''). \quad (20)$$

Hence after renaming several variables in (19), we eventually find combining all with (20):

$$\begin{aligned} [r](X'') = {} & \sum_{y \in S} r_2(y) \cdot \sum_{v \in B} r_1(v) \cdot \big(s_2\big((x = y) \;\&\; v\big) \parallel s_1(y)\big) \cdot [v, y](X'') \\ & + \sum_{w \in B} r_1(w) \cdot \sum_{z \in S} r_2(z) \cdot \big(s_2\big((x = z) \;\&\; w\big) \parallel s_1(z)\big) \cdot [w, z](X''). \end{aligned}$$

Let i be the function in the proof of theorem (2.3.1). Consider the guarded recursive specification E'' after this:

$$\begin{aligned} E'' = \Big\{\; X''_i = {} & \sum_{y \in S} r_2(y) \cdot \sum_{v \in B} r_1(v) \cdot \big(s_2\big((x = y) \;\&\; v\big) \parallel s_1(y)\big) \cdot X''_{i(v,y)} \\ & + \sum_{w \in B} r_1(w) \cdot \sum_{z \in S} r_2(z) \cdot \big(s_2\big((x = z) \;\&\; w\big) \parallel s_1(z)\big) \cdot X''_{i(w,z)} \\ & \;\Big|\; 1 \le i \le |D|^2 \;\Big\}. \end{aligned}$$

Clearly, P'' is a solution for E'', for, if we put for all i:

$$P = X_i = X_{i(v,y)} = X_{i(w,z)}.$$

On the other hand it is immediately clear that, if we put for all i:

$$X''_i = [r_i](X''), \quad X''_{i(v,y)} = [v, y](X''), \quad \text{and } X''_{i(w,z)} = [w, z](X''),$$

this system is also a solution for E''. Therefore we can conclude with the aid of *RSP* that $[r](X'') = P''$. This ends the proof of theorem 2.3.3.

2.4. Prerequisites

In this section we summarize some basic material with which the reader is assumed to be more or less familiar, in case the theoremata below concern closed terms (see, e.g., [**15**]). The difference here is that we prove these basic facts for processes that are solutions of guarded recursive specifications.

Notation (2.4.1)

Let x be the solution of a guarded recursive specification E, then we write $x \in \mathit{GRS}$.

Lemma (2.4.2)

For all $x \in \mathit{GRS}$ the following holds:

$$\forall\, n, k \geq 1, \quad \pi_n \circ \pi_k(x) = \pi_{\min(n,k)}(x).$$

Proof. We will prove the lemma with induction on n. Since $x \in \mathit{GRS}$, we have, in accordance with proposition 5.7 on page 91 in [**7**]:

$$x = \sum_{i=1}^{n} a_i x_i + \sum_{j=1}^{m} b_j, \qquad \text{with } x_i \in \mathit{GRS}.$$

Henceforth, we will use this proposition tacitly. So, if we apply $\pi_1 \circ \pi_k$ to this, we see:

$$\begin{aligned} \pi_1 \circ \pi_k(x) &= \sum_{i=1}^{n} a_i + \sum_{j=1}^{m} b_j \\ &= \pi_{\min(1,k)}(x). \end{aligned}$$

For the moment, take $k = 1$, and $n \geq 1$ then:

$$\begin{aligned} \pi_n \circ \pi_1(x) &= \pi_n\Big(\sum_{i=1}^{n} a_i + \sum_{j=1}^{m} b_j\Big) \\ &= \pi_{\min(n,1)}(x). \end{aligned}$$

Now, let $n \geq 2$, $k \geq 2$, and assume that 2.4.2 is correct up to n inclusive. Then:

$$\begin{aligned} \pi_{n+1} \circ \pi_k(x) &= \sum_{i=1}^{n} a_i \pi_{n-1} \circ \pi_{k-1}(x_i) + \sum_{j=1}^{m} b_j \\ &= \sum_{i=1}^{n} a_i \pi_{\min(n-1,k-1)}(x_i) + \sum_{j=1}^{m} b_j \qquad \text{(induction hypothesis)} \\ &= \sum_{i=1}^{n} \pi_{\min(n-1,k-1)+1}(a_i x_i) + \sum_{j=1}^{m} b_j \\ &= \pi_{\min(n,k)}(x). \end{aligned}$$

This proves 2.4.2.

Proposition (2.4.3)

For all $x, y \in GRS$ the following holds:

$$
\begin{array}{lll}
(i) & & \pi_1(x \mathbin{⫫} y) = \pi_1(x) \\
(ii) & \forall n \geq 1, \forall k \geq n+1, \forall l \geq n: & \pi_{n+1}(x \mathbin{⫫} y) = \pi_{n+1}\big(\pi_k(x) \mathbin{⫫} \pi_l(y)\big) \\
(iii) & \forall n \geq 1, \forall k, l \geq n: & \pi_n(x \mid y) = \pi_n\big(\pi_k(x) \mid \pi_l(y)\big) \\
(iv) & \forall n \geq 1, \forall k, l \geq n: & \pi_n(x \parallel y) = \pi_n\big(\pi_k(x) \parallel \pi_l(y)\big).
\end{array}
$$

Proof. The proof of (i) is trivial. We prove (ii), (iii), and (iv) with induction on n. First take $n = 1$ and $k, l \geq 1$, then:

$$
\begin{aligned}
(ii) \qquad \pi_1\big(\pi_k(x) \mathbin{⫫} \pi_l(y)\big) &\overset{(i)}{=} \pi_1 \circ \pi_k(x) \\
&= \pi_{\min(1,k)}(x) \qquad \text{because of (2.4.2)} \\
&= \pi_1(x) \\
&\overset{(i)}{=} \pi_1(x \mathbin{⫫} y).
\end{aligned}
$$

Since $x, y \in GRS$, we have:

$$
x = \sum_{i=1}^{n} a_i x_i + \sum_{j=1}^{m} b_j = x' + x'', \quad y = \sum_{s=1}^{p} c_s y_s + \sum_{r=1}^{q} d_r = y' + y''.
$$

It is straightforward to see that:

$$
\begin{aligned}
(iii) \qquad \pi_1\big(\pi_k(x) \mid \pi_l(y)\big) &= \sum_{i,r=1}^{n,p} a_i \mid c_r + \sum_{i,s=1}^{n,q} a_i \mid d_s \\
&\quad + \sum_{j,r=1}^{m,p} b_j \mid c_r + \sum_{j,s=1}^{m,q} b_j \mid d_s \\
&= \pi_1(x \mid y).
\end{aligned}
$$

Observe that we proved (ii) only under the assumption that $k, l \geq 1$, so we obtain:

$$
\begin{aligned}
(iv) \qquad \pi_1\big(\pi_k(x) \parallel \pi_l(x)\big) &= \pi_1\big(\pi_k(x) \mathbin{⫫} \pi_l(x)\big) + \pi_1\big(\pi_l(y) \mathbin{⫫} \pi_k(x)\big) \\
&\quad + \pi_1\big(\pi_k(x) \mid \pi_l(x)\big) \\
&= \pi_1\big(\pi_k(x)\big) + \pi_1\big(\pi_l(y)\big) + \pi_1(x \mid y) \\
&= \pi_1(x) + \pi_1(y) + \pi_1(x \mid y) \\
&\overset{(i)}{=} \pi_1(x \mathbin{⫫} y) + \pi_1(y \mathbin{⫫} x) + \pi_1(x \mid y) \\
&= \pi_1(x \parallel y)
\end{aligned}
$$

Now, let $n \geq 1$, and suppose that (*ii*), (*iii*), and (*iv*) have been proved up to n inclusive. We will prove them for $n+1$. Take $k, l \geq n+1$; then we see:

(*iii*)
$$\begin{aligned}\pi_{n+1}\big(\pi_k(x) \mid \pi_l(y)\big) &= \pi_{n+1}\big(\pi_k(x') \mid \pi_l(y')\big)\\ &\quad + \pi_{n+1}\big(\pi_k(x') \mid \pi_l(y'')\big)\\ &\quad + \pi_{n+1}\big(\pi_k(x'') \mid \pi_l(y')\big)\\ &\quad + \pi_{n+1}\big(\pi_k(x'') \mid \pi_l(y'')\big).\end{aligned}$$

We are now going to calculate the four terms of the equation above one at a time:

1
$$\begin{aligned}\pi_{n+1}\big(\pi_k(x') \mid \pi_l(y')\big) &= \pi_{n+1}\Big(\sum_{i=1}^{n} a_i \pi_{k-1}(x_i) \,\Big|\, \sum_{r=1}^{p} \pi_{l-1}(y_j)\Big)\\ &= \sum_{i,r=1}^{n,p} (a_i \mid c_r)\pi_n\big(\pi_{k-1}(x_i) \parallel \pi_{l-1}(y_j)\big)\\ \text{induction hypothesis} \quad &= \sum_{i,r=1}^{n,p} (a_i \mid c_r)\pi_n(x_i \parallel y_j)\\ &= \pi_{n+1}(x' \mid y'),\end{aligned}$$

2
$$\begin{aligned}\pi_{n+1}\big(\pi_k(x') \mid \pi_l(y'')\big) &= \pi_{n+1}\Big(\sum_{i=1}^{n} a_i \pi_{k-1}(x_i) \,\Big|\, \sum_{s=1}^{q} d_s\Big)\\ &= \sum_{i,s=1}^{n,q} (a_i \mid d_s)\pi_n \circ \pi_{k-1}(x_i)\\ &= \sum_{i,s=1}^{n,q} (a_i \mid d_s)\pi_{\min(n,k-1)}(x_i)\\ &= \pi_{n+1}(x' \mid y''),\end{aligned}$$

3
$$\begin{aligned}\pi_{n+1}\big(\pi_k(x'') \mid \pi_l(y')\big) &= \sum_{j,r=1}^{m,p} (b_j \mid c_k)\pi_n \circ \pi_{l-1}(y_r)\\ &= \sum_{j,r=1}^{m,p} (b_j \mid c_k)\pi_n(y_r)\\ &= \pi_{n+1}(x'' \mid y'),\end{aligned}$$

4
$$\begin{aligned}\pi_{n+1}\big(\pi_k(x'') \mid \pi_l(y'')\big) &= \sum_{j,s=1}^{m,q} (b_j \mid d_s)\\ &= \pi_{n+1}(x'' \mid y'').\end{aligned}$$

Combining the equations *1–4*, we obtain the desired result for (*iii*). Now let $k \geq n+2$ and $l \geq n+1$, then we can derive for (*ii*):

$$\begin{aligned}
\pi_{n+1}\big(\pi_k(x) \mathbin{⌊\!⌊} \pi_l(y)\big) &= \pi_{n+1}\Big(\sum_{i=1}^{n} a_i\big(\pi_{k-1}(x_i) \parallel \pi_l(y)\big)\Big) + \sum_{j=1}^{m} b_j \pi_n \circ \pi_l(y) \\
&= \sum_{i=1}^{n} a_i \pi_n(x_i \parallel y) + \sum_{j=1}^{m} b_j \pi_{\min(n,l)}(y) \qquad \text{induction} \\
&= \pi_{n+1}(x \mathbin{⌊\!⌊} y).
\end{aligned}$$

Now we have proved (*ii*) and (*iii*) up to and including $n+1$. So we immediately see that (*iv*) is also valid up to and including $n+1$. This ends the proof of 2.4.3.

Corollary (2.4.4)

For all $x, y \in GRS$ and for all $n \geq 1$, the following holds:

(*i*) $\pi_n(x \mid y) = \pi_n\big(\pi_n(x) \mid \pi_n(y)\big)$

(*ii*) $\pi_n(x \mathbin{⌊\!⌊} y) = \pi_n\big(\pi_n(x) \mathbin{⌊\!⌊} \pi_n(y)\big)$

(*iii*) $\pi_n(x \parallel y) = \pi_n\big(\pi_n(x) \parallel \pi_n(y)\big)$

2.5. The main theorem

In this section we will prove the main theorem of this chapter. We will prove the following. Suppose we have two processes x and y that can be specified with the aid of a guarded recursive specification. Suppose that the registers that occur in x and y are disjoint (this will be formalized with an auxiliary operator later on). Then the register operator $[r]$ distributes over the merge:

$$[r](x \parallel y) = [r](x) \parallel [r](y).$$

In [**4**] we can find a similar theorem stating that the simple state operator distributes in certain circumstances over the merge. In that paper it is demanded that there is no communication, whereas we do not. Unfortunately, the theorem in [**4**] does not hold but it can be repaired. See section 3.4 for a counterexample and a modified version of the theorem.

Before we can state and prove the main theorem we will treat a number of technicalities and we will define the notion of the registers that occur in a process x.

Lemma (2.5.1)

$\forall x \in GRS,\ \forall r \in R,\ \forall n \geq 1\ :\ \pi_n \circ [r](x) = [r] \circ \pi_n(x).$

Proof. It is evident that 2.5.1 holds when x is just an atomic action. Recall that the collection of atomic actions A is of the form:

$$A = A' \cup U(D) \cup U(D\downarrow) \cup U(F) \cup U(F\downarrow).$$

We prove 2.5.1 with induction on n. Since $x \in GRS$ we have, for some $a_i, b_j \in A$:

$$x = \sum_{i=1}^{n} a_i x_i + \sum_{j=1}^{m} b_j \qquad \text{with } x_i \in GRS.$$

Now let $n = 1$, then:

$$\begin{aligned}
\pi_1 \circ [r](x) &= \sum_{i=1}^{n} \pi_1\big([r](a_i)[r_i](x_i)\big) + \sum_{j=1}^{m} \pi_1 \circ [r](b_j) \\
&= \sum_{i=1}^{n} [r](a_i) + \sum_{j=1}^{m} [r] \circ \pi_1(b_j) \\
&= \sum_{i=1}^{n} [r]\big(\pi_1(a_i x_i)\big) + \sum_{j=1}^{m} [r] \circ \pi_1(b_j) \\
&= [r] \circ \pi_1(x),
\end{aligned}$$

for certain $[r_1], \ldots, [r_n] \in R$. Now, suppose that $n > 1$, and 2.5.1 is correct up to n inclusive. Then we can easily see:

$$\begin{aligned}
\pi_{n+1} \circ [r](x) &= \sum_{i=1}^{n} \pi_{n+1}\big([r](a_i)[r_i](x_i)\big) + \sum_{j=1}^{m} \pi_{n+1} \circ [r](b_j) \\
&= \sum_{i=1}^{n} [r](a_i)\pi_n \circ [r_i](x_i) + \sum_{j=1}^{m} [r] \circ \pi_{n+1}(b_j) \\
&= \sum_{i=1}^{n} [r](a_i)[r_i] \circ \pi_n(x_i) + \sum_{j=1}^{m} [r] \circ \pi_{n+1}(b_j) \qquad \text{induction} \\
&= \sum_{i=1}^{n} [r]\big(a_i \pi_n(x_i)\big) + \sum_{j=1}^{m} [r] \circ \pi_{n+1}(b_j) \\
&= \sum_{i=1}^{n} [r] \circ \pi_{n+1}(a_i x_i) + \sum_{j=1}^{m} [r] \circ \pi_{n+1}(b_j) \\
&= [r] \circ \pi_{n+1}(x).
\end{aligned}$$

This proves the Lemma.

Definition (2.5.2)

Consider an element $[r] \in R$. A *register* is a natural number n that stands for the nth position in the register operator $[r]$. Let x be a process in *GRS*. Then we define $\text{reg}(x)$ to be the collection of all the registers that occur in x. The axiomatization of all this is as follows:

$$\begin{aligned}
\text{reg}(a) &= \emptyset && \text{if } a \in A' \cup U(D)\\
\text{reg}(\gamma) &= \emptyset && \text{where } \gamma \text{ is a special constant}\\
\text{reg}\big(u(d \downarrow N_k)\big) &= N_k\\
\text{reg}\big(u\big(f(\uparrow N_k)\big)\big) &= N_k\\
\text{reg}\big(u\big(f(\uparrow N_k) \downarrow M_l\big)\big) &= N_k \cup M_l\\
\text{reg}(\gamma \cdot x) &= \text{reg}(x) && \text{if } \gamma \neq \delta\\
\text{reg}(\delta \cdot x) &= \emptyset\\
\text{reg}(a \cdot x) &= \text{reg}(a) \cup \text{reg}(x) && \text{if } a \neq \delta\\
\text{reg}(x + y) &= \text{reg}(x) \cup \text{reg}(y)
\end{aligned}$$

and for infinite processes we add the following axiom:

$$\text{reg}(x) = \bigcup_{n=1}^{\infty} \text{reg}\big(\pi_n(x)\big)$$

These axioms do not tell us instantaneously how $\text{reg}(x \parallel y)$ has to be evaluated if $x, y \in GRS$. Below we will prove a lemma (see corollary (2.5.6)(*iii*)) that states that $\text{reg}(x \parallel y) \subseteq \text{reg}(x) \cup \text{reg}(y)$. In order to show the lemma, we will first need some intermediate results.

Lemma (2.5.3)

Let $x, y \in GRS$, then the following holds:

$$\forall l \geq 1, \forall k\, (1 \leq k \leq l): \quad \text{reg}\big(\pi_k(x)\big) \subseteq \text{reg}\big(\pi_l(x)\big).$$

Proof. We prove this lemma with induction on l. If $l = 1$, then we have $k = 1$, so there is nothing to prove. Now let $l > 1$, and $1 \leq k \leq l$. As $x \in GRS$, we have that:

$$x = \sum_{i=1}^{n} a_i x_i + \sum_{j=1}^{m} b_j.$$

First we consider the case $k = 1$:

$$\begin{aligned}
\text{reg}\big(\pi_k(x)\big) &= \bigcup_{i=1}^{n} \text{reg}(a_i) \cup \bigcup_{j=1}^{m} \text{reg}(b_j)\\
&\subseteq \bigcup_{i=1}^{n} \Big(\text{reg}(a_i) \cup \text{reg}\big(\pi_{l-1}(x_i)\big)\Big) \cup \bigcup_{j=1}^{m} \text{reg}(b_j)\\
&= \text{reg}\big(\pi_l(x)\big)
\end{aligned}$$

Now let $1 < k \le l$, then we deduce:

$$\begin{aligned}
\operatorname{reg}\bigl(\pi_k(x)\bigr) &= \bigcup_{i=1}^{n} \operatorname{reg}(a_i) \cup \operatorname{reg}\bigl(\pi_{k-1}(x_i)\bigr) \cup \bigcup_{j=1}^{m} \operatorname{reg}(b_j) \\
&\subseteq \bigcup_{i=1}^{n} \operatorname{reg}(a_i) \cup \operatorname{reg}\bigl(\pi_{l-1}(x_i)\bigr) \cup \bigcup_{j=1}^{m} \operatorname{reg}(b_j) \qquad \text{induction} \\
&= \operatorname{reg}\bigl(\pi_l(x)\bigr)
\end{aligned}$$

This finishes the proof of 2.5.3.

Lemma (2.5.4)

$$\begin{array}{lll}
(i) & \forall n \ge 1, \forall k \ge n+1, \forall l \ge n & \operatorname{reg}\bigl(\pi_{n+1}(x \mathbin{⌊\!⌊} y)\bigr) \subseteq \operatorname{reg}\bigl(\pi_k(x)\bigr) \cup \operatorname{reg}\bigl(\pi_l(y)\bigr) \\
(ii) & \forall n \ge 1, \forall k, l \ge n & \operatorname{reg}\bigl(\pi_n(x \mid y)\bigr) \subseteq \operatorname{reg}\bigl(\pi_k(x)\bigr) \cup \operatorname{reg}\bigl(\pi_l(y)\bigr) \\
(iii) & \forall n \ge 1, \forall k, l \ge n & \operatorname{reg}\bigl(\pi_n(x \parallel y)\bigr) \subseteq \operatorname{reg}\bigl(\pi_k(x)\bigr) \cup \operatorname{reg}\bigl(\pi_l(y)\bigr).
\end{array}$$

Proof. We will verify 2.5.4 with induction on n. First we prove (i)–(iii) for $n = 1$. Because of proposition (2.4.3)(i) and lemma (2.5.3), we immediately see:

$$\begin{aligned}
\operatorname{reg}\bigl(\pi_1(x \mathbin{⌊\!⌊} y)\bigr) &= \operatorname{reg}\bigl(\pi_1(x)\bigr) \\
&\subseteq \operatorname{reg}\bigl(\pi_k(x)\bigr) \\
&\subseteq \operatorname{reg}\bigl(\pi_k(x)\bigr) \cup \operatorname{reg}\bigl(\pi_l(x)\bigr)
\end{aligned}$$

First we prove (ii) for atomic actions $a, b \in A$. Observe that (ii) has the ensuing form:

$$\operatorname{reg}(a \mid b) \subseteq \operatorname{reg}(a) \cup \operatorname{reg}(b).$$

If $a \mid b = \delta$, there is nothing to prove. Recall that the set of atoms consists of: $A = A' \cup U(D) \cup U(D\downarrow) \cup U(F) \cup U(F\downarrow)$. If $a \mid b = c$ and $c \in A' \cup U(D)$, then we obviously have $\operatorname{reg}(c) = \emptyset$. So in this case there is nothing to prove either. Now we will treat the verification of the nine remaining possibilities.

1
$$\begin{aligned}
\operatorname{reg}\bigl(u(d) \mid v(d\downarrow N_k)\bigr) &= \operatorname{reg}\bigl(w(d\downarrow N_k)\bigr) \\
&= \emptyset \cup N_k \\
&= \operatorname{reg}\bigl(u(d)\bigr) \cup \operatorname{reg}\bigl(v(d\downarrow N_k)\bigr)
\end{aligned}$$

2
$$\begin{aligned}
\operatorname{reg}\bigl(u(d) \mid v\bigl(f(\uparrow N_k)\bigr)\bigr) &= \operatorname{reg}\bigl(w\bigl((d \wedge f)(\uparrow N_k)\bigr)\bigr) \\
&= \emptyset \cup N_k \\
&= \operatorname{reg}\bigl(u(d)\bigr) \cup \operatorname{reg}\bigl(v\bigl(f(\uparrow N_k)\bigr)\bigr)
\end{aligned}$$

3
$$\begin{aligned}
\operatorname{reg}\bigl(u(d) \mid v\bigl(f(\uparrow N_k)\downarrow M_l\bigr)\bigr) &= \operatorname{reg}\bigl(w\bigl((d \wedge f)(\uparrow N_k)\downarrow M_l\bigr)\bigr) \\
&= \emptyset \cup N_k \cup M_l
\end{aligned}$$

$$
\begin{aligned}
& & &= \operatorname{reg}\big(u(d)\big) \cup \operatorname{reg}\big(v\big(f(\uparrow N_k) \downarrow M_l\big)\big) \\
4 & & \operatorname{reg}\big(u(d \downarrow N_k) \mid v(d \downarrow M_l)\big) &= \operatorname{reg}\big(w(d \downarrow N_k, M_l)\big) \\
& & &= N_k \cup M_l \\
& & &= \operatorname{reg}\big(u(d \downarrow N_k)\big) \cup \operatorname{reg}\big(v(d \downarrow M_l)\big) \\
5 & & \operatorname{reg}\big(u(d \downarrow N_k)\big) \mid v\big(f(\uparrow M_l)\big)\big) &\subseteq \operatorname{reg}\big(w\big((d \wedge f)(\uparrow M_l) \downarrow N_k\big)\big) \\
& & &= N_k \cup M_l \\
& & &= \operatorname{reg}\big(u(d \downarrow N_k)\big) \cup \operatorname{reg}\big(v\big(f(\uparrow M_l)\big)\big) \\
6 & & \operatorname{reg}\big(u(d \downarrow N_k) \mid v\big(f(\uparrow M_l) \downarrow P_n\big)\big) &\subseteq \operatorname{reg}\big(w\big((d \wedge f)(\uparrow M_l) \downarrow N_k, P_n\big)\big) \\
& & &= N_k \cup (M_l \cup P_n) \\
& & &= \operatorname{reg}\big(u(d \downarrow N_k)\big) \\
& & &\quad \cup \operatorname{reg}\big(v\big(f(\uparrow M_l) \downarrow P_n\big)\big)
\end{aligned}
$$

$(S_u = N_k \cup M_l)$:

$$
\begin{aligned}
7 & & \operatorname{reg}\big(u\big(f(\uparrow N_k)\big) \mid v\big(g(\uparrow M_l)\big)\big) &= \operatorname{reg}\big(w\big((f' \wedge g')(\uparrow S_u)\big)\big) \\
& & &= S_u \\
& & &= N_k \cup M_l \\
& & &= \operatorname{reg}\big(u\big(f(\uparrow N_k)\big)\big) \\
& & &\quad \cup \operatorname{reg}\big(v\big(g(\uparrow M_l)\big)\big) \\
8 & & \operatorname{reg}\big(u\big(f(\uparrow N_k)\big) \mid v\big(g(\uparrow M_l) \downarrow P_n\big)\big) &\subseteq \operatorname{reg}\big(w\big((f' \wedge g')(\uparrow S_u) \downarrow P_n\big)\big) \\
& & &= N_k \cup (M_l \cup P_n) \\
& & &= \operatorname{reg}\big(u\big(f(\uparrow N_k)\big)\big) \\
& & &\quad \cup \operatorname{reg}\big(v\big(g(\uparrow M_l) \downarrow P_n\big)\big) \\
9 & & \operatorname{reg}\big(u\big(f(\uparrow N_k) \downarrow P_n\big) \mid v\big(g(\uparrow M_l) \downarrow Q_s\big)\big) &\subseteq \operatorname{reg}\big(w\big((f' \wedge g')(\uparrow S_u) \downarrow P_n, Q_s\big)\big) \\
& & &= (N_k \cup P_n) \cup (M_l \cup Q_s) \\
& & &= \operatorname{reg}\big(u\big(f(\uparrow N_k) \downarrow P_n\big)\big) \\
& & &\quad \cup \operatorname{reg}\big(v\big(g(\uparrow M_l) \downarrow Q_s\big)\big)
\end{aligned}
$$

Now it is also easy to see that (*ii*) is valid if $n = 1$, albeit we still have to distinguish several cases. Because of the fact that $x, y \in GRS$, we have:

$$
x = \sum_{i=1}^{n} a_i x_i + \sum_{j=1}^{m} b_j = x' + x'', \qquad y = \sum_{s=1}^{p} c_s y_s + \sum_{t=1}^{q} d_t = y' + y''.
$$

First we treat the situation that $k = l = 1$.

$$
\operatorname{reg}\big(\pi_1(x \mid y)\big) = \bigcup_{i,s=1}^{n.p} \operatorname{reg}(a_i \mid c_s) \cup \bigcup_{i,t=1}^{n,q} \operatorname{reg}(a_i \mid d_t)
$$

$$\begin{aligned}
&\cup \bigcup_{j,s=1}^{m,p} \operatorname{reg}(b_j \mid c_s) \cup \bigcup_{j,t=1}^{m,q} \operatorname{reg}(b_j \mid d_t)\\
\subseteq &\bigcup_{i,s=1}^{n,p} \big(\operatorname{reg}(a_i) \cup \operatorname{reg}(c_s)\big) \cup \bigcup_{i,t=1}^{n,q} \big(\operatorname{reg}(a_i) \cup \operatorname{reg}(d_t)\big)\\
&\cup \bigcup_{j,s=1}^{n,p} \big(\operatorname{reg}(b_j) \cup \operatorname{reg}(c_s)\big) \cup \bigcup_{j,t=1}^{m,q} \big(\operatorname{reg}(b_j) \cup \operatorname{reg}(d_t)\big)\\
= &\bigcup_{i=1}^{n} \operatorname{reg}(a_i) \cup \bigcup_{j=1}^{m} \operatorname{reg}(b_j) \cup \bigcup_{s=1}^{p} \operatorname{reg}(c_s) \cup \bigcup_{t=1}^{q} \operatorname{reg}(d_t)\\
= &\bigcup_{i=1}^{n} \operatorname{reg}\big(\pi_1(a_i x_i)\big) \cup \bigcup_{j=1}^{m} \operatorname{reg}\big(\pi_1(b_j)\big)\\
&\cup \bigcup_{s=1}^{p} \operatorname{reg}\big(\pi_1(c_s y_s)\big) \cup \bigcup_{t=1}^{q} \operatorname{reg}\big(\pi_1(d_t)\big)\\
= &\operatorname{reg}\big(\pi_1(x)\big) \cup \operatorname{reg}\big(\pi_1(y)\big).
\end{aligned}$$

Now we will handle the case $k, l > 1$. We already saw the first step of the following display:

$$\begin{aligned}
\operatorname{reg}\big(\pi_1(x \mid y)\big) &\subseteq \bigcup_{i=1}^{n} \operatorname{reg}(a_i) \cup \bigcup_{j=1}^{m} \operatorname{reg}(b_j)\\
&\quad \cup \bigcup_{s=1}^{p} \operatorname{reg}(c_s) \cup \bigcup_{t=1}^{q} \operatorname{reg}(d_t)\\
&\subseteq \bigcup_{i=1}^{n} \operatorname{reg}\big(a_i \pi_{k-1}(x_i)\big) \cup \bigcup_{j=1}^{m} \operatorname{reg}\big(\pi_k(b_j)\big)\\
&\quad \cup \bigcup_{s=1}^{p} \operatorname{reg}\big(c_s \pi_{l-1}(y_s)\big) \cup \bigcup_{t=1}^{q} \operatorname{reg}\big(\pi_l(d_t)\big)\\
&= \operatorname{reg}\big(\pi_k(x)\big) \cup \operatorname{reg}\big(\pi_l(y)\big).
\end{aligned}$$

If we have $k = 1$ and $l > 1$ or $k > 1$ and $l = 1$, then we mix the deductions of the former cases to prove the required property. With this we have finished (*ii*). Now it is instantly clear that (*iii*) is valid if $n = 1$, so let $n \geq 1$ and suppose that 2.5.4 has been proved up to and including n. We fix arbitrarily chosen $k \geq n+1$ and $l \geq n$.

$$\operatorname{reg}\big(\pi_{n+1}(x \mathbin{\|\!\!\_} y)\big) = \bigcup_{i=1}^{n} \Big(\operatorname{reg}(a_i) \cup \operatorname{reg}\big(\pi_n(x_i \parallel y)\big)\Big)$$

$$\cup \bigcup_{j=1}^{m} \Big(\operatorname{reg}(b_j) \cup \operatorname{reg}\big(\pi_n(y)\big) \Big)$$

Since $k-1 \geq n$ and $l \geq n$, we may use the induction hypothesis:

$$\begin{aligned}
&\subseteq \bigcup_{i=1}^{n} \Big(\operatorname{reg}(a_i) \cup \operatorname{reg}\big(\pi_{k-1}(x_i)\big) \cup \operatorname{reg}\big(\pi_l(y)\big) \Big) \\
&\qquad \cup \bigcup_{j=1}^{m} \Big(\operatorname{reg}(b_j) \cup \operatorname{reg}\big(\pi_n(y)\big) \Big) \\
&= \operatorname{reg}\big(\pi_k(x)\big) \cup \operatorname{reg}\big(\pi_l(y)\big) \cup \operatorname{reg}\big(\pi_n(y)\big) \qquad \text{use (2.5.3)} \\
&= \operatorname{reg}\big(\pi_k(x)\big) \cup \operatorname{reg}\big(\pi_l(y)\big).
\end{aligned}$$

Now fix $k, l \geq n+1$. Then we infer:

$$\begin{aligned}
\operatorname{reg}\big(\pi_{n+1}(x \mid y)\big) &= \operatorname{reg}\big(\pi_{n+1}(x' \mid y')\big) \cup \operatorname{reg}\big(\pi_{n+1}(x' \mid y'')\big) \\
&\quad \cup \operatorname{reg}\big(\pi_{n+1}(x'' \mid y')\big) \cup \operatorname{reg}\big(\pi_{n+1}(x'' \mid y'')\big).
\end{aligned}$$

We will handle these four terms one at a time:

1 $$\operatorname{reg}\big(\pi_{n+1}(x' \mid y')\big) = \bigcup_{i,s=1}^{n,p} \operatorname{reg}(a_i \mid c_s) \cup \operatorname{reg}\big(\pi_n(x_i \parallel y_s)\big)$$

As $k-1, l-1 \geq n$, we can use induction:

$$\begin{aligned}
&\subseteq \bigcup_{i,s=1}^{n,p} \operatorname{reg}(a_i \mid c_s) \cup \operatorname{reg}\big(\pi_{k-1}(x_i)\big) \cup \operatorname{reg}\big(\pi_{l-1}(y_s)\big) \\
&\subseteq \bigcup_{i,s=1}^{n,p} \operatorname{reg}(a_i) \cup \operatorname{reg}(c_s) \\
&\qquad \cup \operatorname{reg}\big(\pi_{k-1}(x_i)\big) \cup \operatorname{reg}\big(\pi_{l-1}(y_s)\big) \\
&= \operatorname{reg}\big(\pi_k(x')\big) \cup \operatorname{reg}\big(\pi_l(y')\big)
\end{aligned}$$

2 $$\operatorname{reg}\big(\pi_{n+1}(x' \mid y'')\big) = \bigcup_{i,t=1}^{n,q} \operatorname{reg}(a_i \mid d_t) \cup \operatorname{reg}\big(\pi_n(x_i)\big)$$

since $k-1 \geq n$, we can use lemma (2.5.3):

$$\begin{aligned}
&\subseteq \bigcup_{i,t=1}^{n,q} \operatorname{reg}(a_i \mid d_t) \cup \operatorname{reg}\big(\pi_{k-1}(x_i)\big) \\
&\subseteq \bigcup_{i,t=1}^{n,q} \operatorname{reg}(a_i) \cup \operatorname{reg}(d_t) \cup \operatorname{reg}\big(\pi_{k-1}(x_i)\big) \\
&= \bigcup_{i,t=1}^{n,q} \operatorname{reg}(a_i) \cup \operatorname{reg}\big(\pi_l(d_t)\big) \cup \operatorname{reg}\big(\pi_{k-1}(x_i)\big)
\end{aligned}$$

$$= \mathrm{reg}\big(\pi_k(x')\big) \cup \mathrm{reg}\big(\pi_l(y'')\big)$$

3 This case is calculated in precisely the same way as the previous case.

$$\begin{aligned} \textit{4} \quad \mathrm{reg}\big(\pi_{n+1}(x'' \mid y'')\big) &= \bigcup_{j,t=1}^{m,q} \mathrm{reg}(b_j \mid d_t) \\ &\subseteq \bigcup_{j,t=1}^{m,q} \mathrm{reg}(b_j) \cup \mathrm{reg}(d_t) \\ &= \mathrm{reg}\big(\pi_k(x'')\big) \cup \mathrm{reg}\big(\pi_l(y'')\big) \end{aligned}$$

Combining these four terms, we see that (*ii*) is now proved up to and including $n+1$, too; and it will be at once clear that (*iii*) is also valid up to $n+1$ inclusive. This ends the proof of 2.5.4.

Corollary (2.5.5)

For all $x, y \in GRS$ and for all $n \geq 1$, the following holds:

(*i*) $\mathrm{reg}\big(\pi_n(x \mid y)\big) \subseteq \mathrm{reg}\big(\pi_n(x)\big) \cup \mathrm{reg}\big(\pi_n(y)\big)$

(*ii*) $\mathrm{reg}\big(\pi_n(x \mathbin{⌊\!⌊} y)\big) \subseteq \mathrm{reg}\big(\pi_n(x)\big) \cup \mathrm{reg}\big(\pi_n(y)\big)$

(*iii*) $\mathrm{reg}\big(\pi_n(x \parallel y)\big) \subseteq \mathrm{reg}\big(\pi_n(x)\big) \cup \mathrm{reg}\big(\pi_n(y)\big)$

Corollary (2.5.6)

For all $x, y \in GRS$, the following holds:

(*i*) $\mathrm{reg}(x \mid y) \subseteq \mathrm{reg}(x) \cup \mathrm{reg}(y)$

(*ii*) $\mathrm{reg}(x \mathbin{⌊\!⌊} y) \subseteq \mathrm{reg}(x) \cup \mathrm{reg}(y)$

(*iii*) $\mathrm{reg}(x \parallel y) \subseteq \mathrm{reg}(x) \cup \mathrm{reg}(y)$

Proof. Let $\star$ be the communication merge, the left-merge, or the merge. Then we can derive:

$$\begin{aligned} \mathrm{reg}(x \star y) &= \bigcup_{n=1}^{\infty} \mathrm{reg}\big(\pi_n(x \star y)\big) \\ &\subseteq \bigcup_{n=1}^{\infty} \mathrm{reg}\big(\pi_n(x)\big) \cup \mathrm{reg}\big(\pi_n(y)\big) \\ &= \bigcup_{n=1}^{\infty} \mathrm{reg}\big(\pi_n(x)\big) \cup \bigcup_{n=1}^{\infty} \mathrm{reg}\big(\pi_n(y)\big) \\ &= \mathrm{reg}(x) \cup \mathrm{reg}(y). \end{aligned}$$

This concludes the proof of 2.5.6.

Lemma (2.5.7)

Let $x \in GRS$, then the following statements hold:

(*i*) $\quad \operatorname{reg}\big(\partial_H(x)\big) \subseteq \operatorname{reg}(x)$

(*ii*) $\quad \operatorname{reg}\big(\tau_I(x)\big) \subseteq \operatorname{reg}(x)$

Proof. Because of the fact that $\operatorname{reg}(\delta) = \operatorname{reg}(\tau) = \emptyset$, the proof of (*i*) is analogous to the proof of (*ii*). Thence, we omit the proof of (*ii*). Let $x \in GRS$, then we have:

$$x = \sum_{i=1}^{n} a_i x_i + \sum_{j=1}^{m} b_j,$$

so we see:

$$\begin{aligned}
\operatorname{reg}\big(\partial_H(x)\big) &= \bigcup_{a_i \notin H} \big(\operatorname{reg}(a_i) \cup \operatorname{reg}(x_i)\big) \cup \bigcup_{b_j \notin H} \operatorname{reg}(b_j) \\
&\subseteq \bigcup_{i=1}^{n} \big(\operatorname{reg}(a_i) \cup \operatorname{reg}(x_i)\big) \cup \bigcup_{j=1}^{m} \operatorname{reg}(b_j) \\
&= \operatorname{reg}(x).
\end{aligned}$$

Herewith the proof of 2.5.7 is completed.

Proposition (2.5.8)

Let $x \in GRS$, and suppose that $\operatorname{reg}(x) = \{n_1, \ldots, n_k\}$. Let

$$[r] = [d_1, \ldots, d_l] \text{ and } [s] = [e_1, \ldots, e_n]$$

be register operators with $l, n \geq \max\{n_1, \ldots, n_k\}$. If

$$d_{n_i} = e_{n_i} \quad \text{for } 1 \leq i \leq k, \text{ then } [r](x) = [s](x).$$

Proof. It is evident that the statement holds for atomic actions. Nevertheless, as an example we will prove the case $a \in U(F)$:

$$\begin{aligned}
[r]\big(u\big(f(\uparrow M_l)\big)\big) &= u\big(f(d_{m_1}, \ldots, d_{m_l})\big) \\
&= u\big(f(e_{m_1}, \ldots, e_{m_l})\big) \\
\big(\{m_1, \ldots, m_l\} \subseteq \operatorname{reg}(x)\big) \qquad & \\
&= [s]\big(u\big(f(\uparrow M_l)\big)\big)
\end{aligned}$$

Now we are going to prove for all $[r]$ and $[s]$ in R—with the properties mentioned in the proposition—the following with induction on n:

$$\forall n \geq 1, \quad \text{we have} \quad \pi_n \circ [r](x) = \pi_n \circ [s](x). \tag{1}$$

Since $x \in GRS$, we have:

$$x = \sum_{i=1}^{n} a_i x_i + \sum_{j=1}^{m} b_j, \qquad \text{with } x_i \in GRS.$$

First we prove (1) for $n = 1$:

$$\begin{aligned}\pi_1 \circ [r](x) &= \sum_{i=1}^{n} [r](a_i) + \sum_{j=1}^{m} [r](b_j) \\ &= \sum_{i=1}^{n} [s](a_i) + \sum_{j=1}^{m} [s](b_j) \\ &= \pi_1 \circ [s](x).\end{aligned}$$

Now suppose that (1) is true up to n inclusive, then we prove (1) for $n+1$:

On the one hand we can calculate:

$$\begin{aligned}\pi_{n+1} \circ [r](x) &= \sum_{i=1}^{n} [r](a_i)\pi_n \circ [r_i](x_i) + \sum_{j=1}^{m} \pi_{n+1} \circ [r](b_j) \\ &= \sum_{i=1}^{n} [s](a_i)\pi_n \circ [r_i](x_i) + \sum_{j=1}^{m} \pi_{n+1} \circ [s](b_j),\end{aligned}$$

and on the other hand:

$$\begin{aligned}\pi_{n+1} \circ [s](x) &= \sum_{i=1}^{n} \pi_{n+1} \circ [s](a_i x_i) + \sum_{j=1}^{m} \pi_{n+1} \circ [s](b_j) \\ &= \sum_{i=1}^{n} [s](a_i)\pi_n \circ [s_i](x_i) + \sum_{j=1}^{m} \pi_{n+1} \circ [s](b_j),\end{aligned}$$

for certain $[r_i], [s_i] \in R$ $(1 \leq i \leq n)$. Hereinafter, we will verify that these pairs of register operators satisfy the conditions of the proposition. Fix an i with $1 \leq i \leq n$ and let $[r_i] = [d_1^{(i)}, \ldots, d_l^{(i)}]$ and $[s_i] = [e_1^{(i)}, \ldots, e_l^{(i)}]$. Since $\text{reg}(x_i) = \{m_1, \ldots, m_t\} \subseteq \text{reg}(x)$, we have $l, n \geq \max\{m_1, \ldots, m_t\}$. Subsequently we verify the second condition. Choose a j with $1 \leq j \leq t$. If $m_j \notin \text{reg}(a_i)$, then nothing has been changed in this particular position in both $[r_i]$ and $[s_i]$ and then we unequivocally have $d_{m_j}^{(i)} = e_{m_j}^{(i)}$. So assume that $m_j \in \text{reg}(a_i)$. Even in this case there might be no changes at all, e.g., if $a_i = r\big(id(\uparrow m_j)\big)$. And we obtain again $d_{m_j}^{(i)} = e_{m_j}^{(i)}$. Now let us additionally suppose that d_{m_j} has been changed to $d' = d_{m_j}^{(i)}$. Then e_{m_j} has been changed as well to $d' = e_{m_j}^{(i)}$. So in this case, too, we obtain equality. Hence $[r_i]$ and $[s_i]$ satisfy the required conditions. Now we can use the induction hypothesis and we finally obtain $[r_i](x_i) = [s_i](x_i)$. This proves (1), so with the aid of *AIP* we conclude the proof of 2.5.8.

Theorem (2.5.9)

Let x, y be closed ACP-terms. If $\mathrm{reg}(x) \cap \mathrm{reg}(y) = \emptyset$, then the following holds for all $[r] \in R$:

(*i*) $[r](x \mathbin{\lfloor\!\lfloor} y) = [r](x) \mathbin{\lfloor\!\lfloor} [r](y)$,

(*ii*) $[r](x \mid y) = [r](x) \mid [r](y)$,

(*iii*) $[r](x \parallel y) = [r](x) \parallel [r](y)$.

Proof. We prove this theorem with induction on the sum n of the number of symbols of x and y. So let us first consider the basis of our induction: the case $n = 2$. Then $x, y \in A$.

$$
\begin{array}{lrl}
(i) & [r](x \mathbin{\lfloor\!\lfloor} y) & = [r](x \cdot y) \\
\text{for a certain } [r'] \in R & & = [r](x)[r'](y) \\
(2) & & = [r](x)[r](y) \\
 & & = [r](x) \mathbin{\lfloor\!\lfloor} [r](y).
\end{array}
$$

Equality (2) is obtained by proposition (2.5.8) and the fact that

$$\mathrm{reg}(x) \cap \mathrm{reg}(y) = \emptyset.$$

For, we can freely change every register of $[r']$ that does not occur in $\mathrm{reg}(y)$. But the only registers that could have been changed are registers in $\mathrm{reg}(x)$. This means that we can use proposition (2.5.8) and we find: $[r](y) = [r'](y)$.

Now we are going to prove the second claim of 2.5.9 for atomic actions. Recall that we have several different atomic actions, so we have a lot of cases. Even if we use the fact that for closed terms t and s we have $s \mid t = t \mid s$ there are still 15 possibilities. We choose an x five times, and we vary y for all the cases:

1 $x = a \in A'$

1.1 $[r](a \mid b) = [r](\delta) = a \mid b = [r](a) \mid [r](b)$.

1.2 $[r]\big(a \mid u(d)\big) = [r](\delta) = a \mid u(d) = [r](a) \mid [r]\big(u(d)\big)$

1.3 $[r]\big(a \mid u(d \downarrow N_k)\big) = \delta = a \mid u(d) = [r](a) \mid [r]\big(u(d \downarrow N_k)\big)$

1.4 $[r]\big(a \mid u\big(f(\uparrow N_k)\big)\big) = \delta = a \mid u\big(f(d_{n_1}, \ldots, d_{n_k})\big) = [r](a) \mid [r]\big(u\big(f(\uparrow N_k)\big)\big)$

1.5 $[r]\big(a \mid u\big(f(\uparrow N_k) \downarrow M_l\big)\big) = \delta = a \mid u\big(f(d_{n_1}, \ldots, d_{n_k})\big)$

$= [r](a) \mid [r]\big(u\big(f(\uparrow N_k) \downarrow M_l\big)\big)$

2 $x = u(d) \in U(D)$

2.1 $y \in U(D)$

2.1.1 $[r]\big(u(d) \mid v(e)\big) = \delta = u(d) \mid v(e) = [r]\big(u(d)\big) \mid [r]\big(v(e)\big)$

2.1.2 $[r]\big(u(d) \mid v(d)\big) = [r]\big(w(d)\big) = u(d) \mid v(d) = [r]\big(u(d)\big) \mid [r]\big(v(d)\big)$

We have seen that we subdivided *2.1* here. Henceforth, we will omit the first subcase (*2.1.1*).

2.2 $[r]\big(u(d) \mid v(d \downarrow N_k)\big) = [r]\big(w(d \downarrow N_k)\big) = w(d) = u(d) \mid v(d)$
$= [r]\big(u(d)\big) \mid [r]\big(v(d \downarrow N_k)\big)$

2.3 $[r]\big(u(d) \mid v\big(f(\uparrow N_k)\big)\big) = [r]\big(w\big((d \wedge f)(\uparrow N_k)\big)\big)$

$$= \begin{cases} w(d), & \text{if } d = f(d_{n_1}, \dots d_{n_k}) \\ \delta, & \text{otherwise} \end{cases}$$

$= u(d) \mid v\big(f(d_{n_1}, \dots d_{n_k})\big) = [r]\big(u(d)\big) \mid [r]\big(v\big(f(\uparrow N_k)\big)\big)$

Notice that, because of $\operatorname{reg}(x) \cap \operatorname{reg}(y) = \emptyset$, the communications satisfy the register conditions (see section 2.2, page 25).

2.4 $[r]\big(u(d) \mid v\big(f(\uparrow N_k) \downarrow M_l\big)\big) = [r]\big(w\big((d \wedge f)(\uparrow N_k) \downarrow M_l\big)\big)$

$$= \begin{cases} w(d), & \text{if } d = f(d_{n_1}, \dots d_{n_k}) \\ \delta, & \text{otherwise} \end{cases}$$

$= u(d) \mid v\big(f(d_{n_1}, \dots d_{n_k})\big) = [r]\big(u(d)\big) \mid [r]\big(v\big(f(\uparrow N_k) \downarrow M_l\big)\big)$

3 $x = u(d \downarrow N_k) \in U(D \downarrow)$

3.1 $[r]\big(u(d \downarrow N_k) \mid v(d \downarrow M_l)\big) = [r]\big(w(d \downarrow N_k, M_l)\big)$
$= w(d) = u(d) \mid v(d) = [r]\big(u(d \downarrow N_k)\big) \mid [r]\big(v(d \downarrow M_l)\big)$

3.2 $[r]\big(u(d \downarrow N_k) \mid v\big(f(\uparrow M_l)\big)\big) = [r]\big(w\big((d \wedge f)(\uparrow M_l) \downarrow N_k\big)\big)$

$$= \begin{cases} w(d), & \text{if } d = f(d_{n_1}, \dots, d_{n_k}) \\ \delta, & \text{otherwise.} \end{cases}$$

$= u(d) \mid v\big(f(d_{n_1}, \dots, d_{n_k})\big) = [r]\big(u(d \downarrow N_k)\big) \mid [r]\big(v\big(f(\uparrow M_l)\big)\big)$

3.3 $[r]\big(u(d \downarrow N_k) \mid v\big(f(\uparrow M_l) \downarrow P_n\big)\big) = [r]\big(w\big((d \wedge f)(\uparrow M_l) \downarrow N_k, P_n\big)\big)$

$$= \begin{cases} w(d), & \text{if } d = f(d_{n_1}, \dots, d_{n_k}) \\ \delta, & \text{otherwise.} \end{cases}$$

$= u(d) \mid v\big(f(d_{n_1}, \dots, d_{n_k})\big) = [r]\big(u(d \downarrow N_k)\big) \mid [r]\big(v\big(f(\uparrow M_l) \downarrow P_n\big)\big)$

4 $x = u\big(f(\uparrow N_k)\big) \in U(F)$

4.1 $[r]\big(u\big(f(\uparrow N_k)\big) \mid v\big(g(\uparrow M_l)\big)\big) = [r]\big(w\big((f' \wedge g')(\uparrow S_u)\big)\big)$

$$= \begin{cases} w\big(f(d_{n_1}, \dots, d_{n_k})\big), & \text{if } f(d_{n_1}, \dots, d_{n_k}) = g(d_{m_1}, \dots, d_{m_l}) \\ \delta, & \text{otherwise.} \end{cases}$$

$= u\big(f(d_{n_1}, \dots, d_{n_k})\big) \mid v\big(g(d_{m_1}, \dots, d_{m_l})\big)$
$= [r]\big(u\big(f(\uparrow N_k)\big)\big) \mid [r]\big(v\big(g(\uparrow M_l)\big)\big)$

4.2 $[r]\big(u\big(f(\uparrow N_k)\big) \mid v\big(g(\uparrow M_l) \downarrow P_n\big)\big) = [r]\big(w\big((f' \wedge g')(\uparrow S_u) \downarrow P_n\big)\big)$

$$= \begin{cases} w\big(f(d_{n_1}, \ldots, d_{n_k})\big), & \text{if } f(d_{n_1}, \ldots, d_{n_k}) = g(d_{m_1}, \ldots, d_{m_l}) \\ \delta, & \text{otherwise.} \end{cases}$$

$$= u\big(f(d_{n_1}, \ldots, d_{n_k})\big) \mid v\big(g(d_{m_1}, \ldots, d_{m_l})\big)$$

$$= [r]\big(u\big(f(\uparrow N_k)\big)\big) \mid [r]\big(v\big(g(\uparrow M_l) \downarrow P_n\big)\big)$$

5 $x = u\big(f(\uparrow N_k) \downarrow P_n\big) \in U(F)$

5.1 $[r]\big(u\big(f(\uparrow N_k) \downarrow P_n\big) \mid v\big(g(\uparrow M_l) \downarrow Q_s\big)\big) = [r]\big(w\big((f' \wedge g')(\uparrow S_u) \downarrow P_n, Q_s\big)\big)$

$$= \begin{cases} w\big(f(d_{n_1}, \ldots, d_{n_k})\big), & \text{if } f(d_{n_1}, \ldots, d_{n_k}) = g(d_{m_1}, \ldots, d_{m_l}) \\ \delta, & \text{otherwise.} \end{cases}$$

$$= u\big(f(d_{n_1}, \ldots, d_{n_k})\big) \mid v\big(g(d_{m_1}, \ldots, d_{m_l})\big)$$

$$= [r]\big(u\big(f(\uparrow N_k) \downarrow P_n\big)\big) \mid [r]\big(v\big(g(\uparrow M_l) \downarrow Q_s\big)\big)$$

Now we have exhaustively seen that (*ii*) holds for atomic actions. We already proved that (*i*) is valid for atomic actions, so it is immediately clear that (*iii*) also holds for atoms. Hence assume that 2.5.9 has been proved for closed terms x and y with the sum of the number of symbols less than or equal to n. Choose now closed terms x and y with the sum of the number of symbols equal to $n + 1$. Since x and y are closed terms we have:

$$x = \sum_{i=1}^{n} a_i x_i + \sum_{j=1}^{m} b_j = x' + x'', \quad y = \sum_{k=1}^{p} c_k y_k + \sum_{l=1}^{q} d_l = y' + y''.$$

for certain closed terms $x_1, \ldots, x_n, y_1, \ldots, y_p$. Now we calculate $[r](x \mathbin{⌊\!⌊} y)$:

$$\begin{aligned}
[r](x \mathbin{⌊\!⌊} y) &= \sum_{i=1}^{n} [r](a_i)[r_i](x_i \parallel y) + \sum_{j=1}^{m} [r](b_j)[r_j](y) \\
\text{(induction)} \qquad &= \sum_{i=1}^{n} [r](a_i) \cdot \big([r_i](x_i) \parallel [r_i](y)\big) + \sum_{j=1}^{m} [r](b_j) \mathbin{⌊\!⌊} [r_j](y) \\
&= \sum_{i=1}^{n} [r](a_i)[r_i](x_i) \mathbin{⌊\!⌊} [r_i](y) + \sum_{j=1}^{m} [r](b_j) \mathbin{⌊\!⌊} [r_j](y) \\
&= \sum_{i=1}^{n} [r](a_i x_i) \mathbin{⌊\!⌊} [r_i](y) + \sum_{j=1}^{m} [r](b_j) \mathbin{⌊\!⌊} [r_j](y) \\
(3) \qquad &= \sum_{i=1}^{n} [r](a_i x_i) \mathbin{⌊\!⌊} [r](y) + \sum_{j=1}^{m} [r](b_j) \mathbin{⌊\!⌊} [r](y) \\
&= [r](x) \mathbin{⌊\!⌊} [r](y).
\end{aligned}$$

Notice that we have changed $[r_i]$ to $[r]$ in the first term of (3), and $[r_j]$ to $[r]$ in the second term. We can this do because of:

$$\mathrm{reg}(a_i) \cap \mathrm{reg}(y) \subseteq \mathrm{reg}(x) \cap \mathrm{reg}(y) = \emptyset, \qquad 1 \le i \le n$$

and

$$\mathrm{reg}(b_j) \cap \mathrm{reg}(y) \subseteq \mathrm{reg}(x) \cap \mathrm{reg}(y) = \emptyset \qquad 1 \le j \le m,$$

and proposition (2.5.8). In the sequel we will tacitly change registers as explained above. Now we calculate $[r](x \mid y)$:

$$[r](x \mid y) = [r](x' \mid y') + [r](x' \mid y'') + [r](x'' \mid y') + [r](x'' \mid y'').$$

Below we will calculate these four terms one at a time:

1
$$[r](x' \mid y') = \sum_{i,k=1}^{n,p} [r](a_i \mid c_k)[r_{i,k}](x_i \parallel y_k)$$

(induction)
$$= \sum_{i,k=1}^{n,p} [r](a_i \mid c_k)\big([r_{i,k}](x_i) \parallel [r_{i,k}](y_k)\big)$$
$$= \sum_{i,k=1}^{n,p} \big([r](a_i) \mid [r](c_k)\big)\big([r_{i,k}](x_i) \parallel [r_{i,k}](y_k)\big)$$
$$= \sum_{i,k=1}^{n,p} [r](a_i)[r_{i,k}](x_i) \mid [r](c_k)[r_{i,k}](y_k)$$
$$= \sum_{i,k=1}^{n,p} [r](a_i)[r_i](x_i) \mid [r](c_k)[r_k](y_k)$$
$$= [r](x') \mid [r](y').$$

2
$$[r](x' \mid y'') = \sum_{i,l=1}^{n,q} [r](a_i \mid d_l)[r_{i,l}](x_i)$$
$$= \sum_{i,l=1}^{n,q} [r](a_i \mid d_l)[r_i](x_i)$$
$$= \sum_{i,l=1}^{n,q} \big([r](a_i) \mid [r](d_l)\big)[r_i](x_i)$$
$$= \sum_{i,l=1}^{n,q} [r](a_i x_i) \mid [r](d_l)$$
$$= [r](x') \mid [r](y'').$$

3
$$[r](x'' \mid y') = \sum_{j,k=1}^{m,p} [r](b_j \mid c_k)[r_{j,k}](y_k)$$

$$\begin{aligned}
&= \sum_{j,k=1}^{m,p} [r](b_j \mid c_k)[r_k](y_k) \\
&= \sum_{j,k=1}^{m,p} \big([r](b_j) \mid [r](c_k)\big)[r_k](y_k) \\
&= \sum_{j,k=1}^{m,p} [r](b_j) \mid [r](c_k y_k) \\
&= [r](x'') \mid [r](y').
\end{aligned}$$

4 $$\begin{aligned}
[r](x'' \mid y'') &= \sum_{j,l=1}^{m,q} [r](b_j \mid d_l) \\
&= \sum_{j,l=1}^{m,q} [r](b_j) \mid [r](d_l) \\
&= [r](x'') \mid [r](y'').
\end{aligned}$$

If we combine these four equations we immediately see that (ii) is correct up to and including $n+1$. We also proved that (i) is valid up to $n+1$ inclusive, thus combining these results we obtain that (iii) is true up to $n+1$ inclusive. This ends the proof of 2.5.9.

Lemma (2.5.10)

If $x, y \in GRS$ then we have: $x \parallel y \in GRS$.

Proof. If we combine theorem 2.14 and proposition 2.15 in [**44**], the lemma immediately follows. This finishes 2.5.10.

The main theorem (2.5.11)

Let $n \geq 1$. If we have for $x_1, \dots, x_n \in GRS$:

$$\forall i, j : 1 \leq i < j \leq n : \quad \mathrm{reg}(x_i) \cap \mathrm{reg}(x_j) = \emptyset,$$

then $\forall [r] \in R : [r](x_1 \parallel x_2 \parallel \cdots \parallel x_n) = [r](x_1) \parallel [r](x_2) \parallel \cdots \parallel [r](x_n)$.

Proof. We prove 2.5.11 with induction on n. Let us first consider the case $n = 2$: for x and y with the properties mentioned in 2.5.11, we are to show $[r](x \parallel y) = [r](x) \parallel [r](y)$. In order to prove this, it suffices to prove:

$$\forall k \geq 1 : \pi_k \circ [r](x \parallel y) = \pi_k\big([r](x) \parallel [r](y)\big), \tag{4}$$

because of AIP. We now will prove (4) below:

$$\begin{aligned}
\pi_k \circ [r](x \parallel y) &= [r] \circ \pi_k(x \parallel y) && (2.5.1) \\
&= [r] \circ \pi_k\big(\pi_k(x) \parallel \pi_k(y)\big) && (2.4.4)(iii) \\
&= \pi_k \circ [r]\big(\pi_k(x) \parallel \pi_k(y)\big) && (2.5.1)
\end{aligned}$$

Observe that $\pi_k(x)$ and $\pi_k(y)$ are closed terms, so we can use here (2.5.9)

$$\begin{aligned} &= \pi_k\big([r] \circ \pi_k(x) \parallel [r] \circ \pi_k(y)\big) \\ &= \pi_k\big(\pi_k \circ [r](x) \parallel \pi_k \circ [r](y)\big) && (2.5.1) \\ &= \pi_k\big([r](x) \parallel [r](y)\big). && (2.4.4)(iii) \end{aligned}$$

We now finished the case $n = 2$. Suppose that 2.5.11 has been proved up to n inclusive, then we prove it for $n + 1$:

$$\forall 1 \leq i \leq n : \mathrm{reg}(x_i) \cap \mathrm{reg}(x_{n+1}) = \emptyset \Longrightarrow \Big(\bigcup_{i=1}^{n} \mathrm{reg}(x_i)\Big) \cap \mathrm{reg}(x_{n+1}) = \emptyset.$$

Since, with induction, it follows easily from lemma (2.5.10), that the merge of n processes, that are in *GRS*, is itself in *GRS*, we can use the induction hypothesis below:

$$\begin{aligned} [r](x_1 \parallel \cdots \parallel x_{n+1}) &= [r]\big((x_1 \parallel \cdots \parallel x_n) \parallel x_{n+1}\big) \\ &= [r](x_1 \parallel \cdots \parallel x_n) \parallel [r](x_{n+1}) && (n = 2) \\ &= \big([r](x_1) \parallel \cdots \parallel [r](x_n)\big) \parallel [r](x_{n+1}) && \text{induction} \\ &= [r](x_1) \parallel \cdots \parallel [r](x_{n+1}). \end{aligned}$$

This concludes the proof of 2.5.11.

At this point we need another result which states that if we alter the positions of the registers in a process and simultaneously alter the contents of the register operator, then we want this process to behave exactly the same. As an example, consider the following process: $[d]\big(r(\uparrow 1)\big)$. Now we alter the position of the register to the second position, and we also put the d on this position: $[d', d]\big(r(\uparrow 2)\big)$. If we eliminate the register operator in the first expression, we obtain: $r(d)$. If we do the same with the second expression then we derive the same result. Below we will axiomatize these transformations on registers operators and processes.

Definition (2.5.12)

Let $x \in GRS$. Suppose that we have a bijection σ between $\mathrm{reg}(x)$ and $M \subset \mathbb{N}$. Define a function $h_\sigma : A \longrightarrow A \cup C$ as follows:

$$\begin{aligned} h_\sigma\big(u(d \downarrow N_k)\big) &= u\big(d \downarrow \sigma(N_k)\big) \\ h_\sigma\big(u\big(f(\uparrow N_k)\big)\big) &= u\big(f\big(\uparrow \sigma(N_k)\big)\big) \\ h_\sigma\big(u\big(f(\uparrow N_k) \downarrow M_l\big)\big) &= u\big(f\big(\uparrow \sigma(N_k)\big) \downarrow \sigma(M_l)\big), \end{aligned}$$

and for all the other atomic actions:

$$h_\sigma(a) = a.$$

Now we define ρ_{h_σ} to be the renaming belonging to h_σ. For the sake of convenience, we will enumerate hereinafter the relevant axioms for ρ_{h_σ}. The definition of a renaming operator can also be found in (1.3.1); see page 15.

$$\begin{aligned}
\rho_{h_\sigma}(\gamma) &= \gamma \\
\rho_{h_\sigma}(a) &= h_\sigma(a) \\
\rho_{h_\sigma}(x \cdot y) &= \rho_{h_\sigma}(x) \cdot \rho_{h_\sigma}(y) \\
\rho_{h_\sigma}(x + y) &= \rho_{h_\sigma}(x) + \rho_{h_\sigma}(y)
\end{aligned}$$

Where $a \in A$, γ is a special constant, and x, y are processes.

Lemma (2.5.13)

Let ρ_{h_σ} be as in definition (2.5.12) above, then the following holds:

$$\forall n \in \mathbb{N},\ \forall x \in GRS: \quad \rho_{h_\sigma} \circ \pi_n(x) = \pi_n \circ \rho_{h_\sigma}(x).$$

Proof. We will prove this statement with induction on n. First consider the case $n = 1$:

$$\begin{aligned}
\text{recall that} \quad x &= \sum_{i=1}^{n} a_i x_i + \sum_{j=1}^{m} b_j \\
\rho_{h_\sigma} \circ \pi_1(x) &= \rho_{h_\sigma} \circ \pi_1\Big(\sum_{i=1}^{n} a_i x_i + \sum_{j=1}^{m} b_j\Big) \\
&= \sum_{i=1}^{n} \rho_{h_\sigma}(a_i) + \sum_{j=1}^{m} \rho_{h_\sigma}(b_j) \\
&= \sum_{i=1}^{n} \pi_1\big(\rho_{h_\sigma}(a_i) \cdot \rho_{h_\sigma}(x_i)\big) + \sum_{j=1}^{m} \pi_1 \circ \rho_{h_\sigma}(b_j) \\
&= \pi_1 \circ \rho_{h_\sigma}(x).
\end{aligned}$$

Now assume 2.5.13 is correct up to n inclusive, then we prove it for $n + 1$:

$$\begin{aligned}
\rho_{h_\sigma} \circ \pi_{n+1}(x) &= \sum_{i=1}^{n} \rho_{h_\sigma}(a_i) \cdot \rho_{h_\sigma} \circ \pi_n(x_i) + \sum_{j=1}^{m} \rho_{h_\sigma}(b_j) \\
&= \sum_{i=1}^{n} \rho_{h_\sigma}(a_i) \cdot \pi_n \circ \rho_{h_\sigma}(x_i) + \sum_{j=1}^{m} \pi_{n+1} \circ \rho_{h_\sigma}(b_j) \\
&= \pi_{n+1} \circ \rho_{h_\sigma}(x).
\end{aligned}$$

This ends the verification of 2.5.13.

Proposition (2.5.14)

We use here the notations of definition (2.5.12). Let $x \in GRS$. Let

$$l \geq \max\{y : y \in \mathrm{reg}(x)\}, n \geq \max\{y : y \in M\}.$$

Let $[r] = [d_1, \ldots, d_l]$, $[s] = [e_1, \ldots, e_n] \in R$. Then we have:

$$\forall y \in \mathrm{reg}(x) : \quad d_y = e_{\sigma(y)} \Longrightarrow [r](x) = [s] \circ \rho_{h_\sigma}(x).$$

Proof. First we show that 2.5.14 is correct for atomic actions: If $a \in A' \cup U(D)$, then there is nothing to prove. Now we will handle the three remaining cases:

$$\begin{aligned}
[r]\big(u(d \downarrow N_k)\big) &= u(d) \\
&= [s]\big(u\big(d \downarrow \sigma(N_k)\big)\big) \\
&= [s] \circ \rho_{h_\sigma}\big(u(d \downarrow N_k)\big) \\
[r]\big(u\big(f(\uparrow N_k)\big)\big) &= u\big(f(d_{n_1}, \ldots, d_{n_k})\big) \\
&= u\big(f(d_{\sigma(n_1)}, \ldots, d_{\sigma(n_k)})\big) \\
&= [s]\big(u\big(f\big(\uparrow \sigma(N_k)\big)\big)\big) \\
&= [s] \circ \rho_{h_\sigma}\big(u\big(f(\uparrow N_k)\big)\big) \\
[r]\big(u\big(f(\uparrow N_k) \downarrow M_l\big)\big) &= u\big(f(d_{n_1}, \ldots, d_{n_k})\big) \\
&= u\big(f(d_{\sigma(n_1)}, \ldots, d_{\sigma(n_k)})\big) \\
&= [s]\big(u\big(f\big(\uparrow \sigma(N_k)\big) \downarrow \sigma(M_l)\big)\big) \\
&= [s] \circ \rho_{h_\sigma}\big(u\big(f(\uparrow N_k) \downarrow M_l\big)\big).
\end{aligned}$$

At this point, we will prove 2.5.14 for closed terms. We prove the statement with induction on the number of symbols of $x = \sum_{i=1}^{n} a_i x_i + \sum_{j=1}^{m} b_j$.

On the one hand we have:

$$[r](x) = \sum_{i=1}^{n} [r](a_i)[r_i](x_i) + \sum_{j=1}^{m} [r](b_j),$$

and on the other hand we have:

$$\begin{aligned}
[s] \circ \rho_{h_\sigma}(x) &= \sum_{i=1}^{n} [s] \circ \rho_{h_\sigma}(a_i)[s_i] \circ \rho_{h_\sigma}(x_i) + \sum_{j=1}^{m} [s] \circ \rho_{h_\sigma}(b_j) \\
&= \sum_{i=1}^{n} [r](a_i)[s_i] \circ \rho_{h_\sigma}(x_i) + \sum_{j=1}^{m} [r](b_j)
\end{aligned}$$

Write $[r_i] = [d_1^{(i)}, \ldots, d_l^{(i)}]$ and $[s_i] = [e_1^{(i)}, \ldots, e_n^{(i)}]$. Now we show the following:

$$\forall y \in \mathrm{reg}(x_i) : \quad d_y^{(i)} = e_{\sigma(y)}^{(i)}.$$

If $y \notin \mathrm{reg}(a_i)$, then there is nothing to prove. Thus assume that $y \in \mathrm{reg}(a_i)$. Even in this situation there might be no changes at all, e.g., if only a register is read. So let us moreover assume that d_y is changed to $d' = d_y^{(i)}$, then $e_{\sigma(y)}$ is changed to $d' = e_{\sigma(y)}^{(i)}$ as well. This means that we may use the induction hypothesis and we see:

$$[r_i](x_i) = [s_i] \circ \rho_{h_\sigma}(x_i).$$

Finally, let $x \in GRS$. Fix an $n \in \mathbb{N}$, then it suffices to make the following calculation:

$$\begin{aligned}
\pi_n \circ [r](x) &= [r] \circ \pi_n(x) && (2.5.1)\\
&= [s] \circ \rho_{h_\sigma} \circ \pi_n(x)\\
&= [s] \circ \pi_n \circ \rho_{h_\sigma}(x) && (2.5.13)\\
&= \pi_n \circ [s] \circ \rho_{h_\sigma}(x). && (2.5.1)
\end{aligned}$$

Herewith we ended the proof of the proposition.

Theorem (2.5.15)

Let notations be as in definition (2.5.12). For all $x, y \in GRS$ the following holds:

(*i*) $\rho_{h_\sigma}(x \mathbin{⌊\!⌊} y) = \rho_{h_\sigma}(x) \mathbin{⌊\!⌊} \rho_{h_\sigma}(y)$

(*ii*) $\rho_{h_\sigma}(x \mid y) = \rho_{h_\sigma}(x) \mid \rho_{h_\sigma}(y)$

(*iii*) $\rho_{h_\sigma}(x \parallel y) = \rho_{h_\sigma}(x) \parallel \rho_{h_\sigma}(y)$.

Proof. First we prove 2.5.15 for closed terms. This we do with induction on the sum of the number of symbols of x and y. The proof of (i) for atomic actions is trivial. The same applies to the proof of (ii) for atomic actions, albeit we have to distinguish a lot of cases. Due to the fact that ρ_{h_σ} distributes over the alternative composition, the proof of (iii)—for atomic actions—is simple, too. Now we prove the induction step for (i):

$$\begin{aligned}
\rho_{h_\sigma}(x \mathbin{⌊\!⌊} y) &= \sum_{i=1}^{n} \rho_{h_\sigma}(a_i) \cdot \rho_{h_\sigma}(x_i \parallel y) + \sum_{j=1}^{m} \rho_{h_\sigma}(b_j) \cdot \rho_{h_\sigma}(y)\\
&= \sum_{i=1}^{n} \rho_{h_\sigma}(a_i) \cdot \big(\rho_{h_\sigma}(x_i) \parallel \rho_{h_\sigma}(y)\big) + \sum_{j=1}^{m} \rho_{h_\sigma}(b_j) \mathbin{⌊\!⌊} \rho_{h_\sigma}(y)\\
&= \sum_{i=1}^{n} \rho_{h_\sigma}(a_i x_i) \mathbin{⌊\!⌊} \rho_{h_\sigma}(y) + \sum_{j=1}^{m} \rho_{h_\sigma}(b_j) \mathbin{⌊\!⌊} \rho_{h_\sigma}(y)\\
&= \rho_{h_\sigma}(x) \mathbin{⌊\!⌊} \rho_{h_\sigma}(y).
\end{aligned}$$

The verification of (ii):

$$\rho_{h_\sigma}(x \mid y) = \rho_{h_\sigma}(x' \mid y') + \rho_{h_\sigma}(x' \mid y'') + \rho_{h_\sigma}(x'' \mid y') + \rho_{h_\sigma}(x'' \mid y'')$$

As an example we treat the case $\rho_{h_\sigma}(x' \mid y')$:

$$\begin{aligned}
\rho_{h_\sigma}(x' \mid y') &= \sum_{i,s=1}^{n,p} \rho_{h_\sigma}(a_i \mid c_s) \cdot \rho_{h_\sigma}(x_i \parallel y_s) \\
&= \sum_{i,s=1}^{n,p} \big(\rho_{h_\sigma}(a_i) \mid \rho_{h_\sigma}(c_s)\big) \cdot \big(\rho_{h_\sigma}(x_i) \parallel \rho_{h_\sigma}(y_s)\big) \\
&= \sum_{i,s=1}^{n,p} \rho_{h_\sigma}(a_i x_i) \mid \rho_{h_\sigma}(c_s y_s) \\
&= \rho_{h_\sigma}(x') \mid \rho_{h_\sigma}(y').
\end{aligned}$$

Using the aforecited distributivity again, the proof of (iii) is obvious. Now we drop the assumption about x and y being closed terms. Let $\star$ be the merge, the left-merge, or the communication merge, then we calculate for each $n \geq 1$:

$$\begin{aligned}
\pi_n \circ \rho_{h_\sigma}(x \star y) &= \rho_{h_\sigma} \circ \pi_n(x \star y) && (2.5.13) \\
&= \rho_{h_\sigma} \circ \pi_n\big(\pi_n(x) \star \pi_n(y)\big) && (2.4.4) \\
&= \pi_n \circ \rho_{h_\sigma}\big(\pi_n(x) \star \pi_n(y)\big) && (2.5.13)
\end{aligned}$$

Recall that $\pi_n(x)$ and $\pi_n(y)$ are closed terms, so we can apply 2.5.15.

$$\begin{aligned}
&= \pi_n\big(\rho_{h_\sigma} \circ \pi_n(x) \star \rho_{h_\sigma} \circ \pi_n(y)\big) \\
&= \pi_n\big(\pi_n \circ \rho_{h_\sigma}(x) \star \pi_n \circ \rho_{h_\sigma}(y)\big) && (2.5.13) \\
&= \pi_n\big(\rho_{h_\sigma}(x) \star \rho_{h_\sigma}(y)\big) && (2.4.4)
\end{aligned}$$

With the aid of *AIP*, we conclude this proof.

2.6. An Application

As an application of the theory that we treated so far, we discuss in this section an elaborate example. We will give a formal proof of a theorem on palindrome recognition (see section 5.1 theorem 5.1 in [**44**]). We do not intend to explain the theory of systolic arrays, nor the intuition behind the specifications that will be given hereinafter. We merely see the proof of this theorem as an application of the register operator, not the implications of this theorem. For a full explanation of the implications we advise the reader to read section 5.1 of [**44**].

Here is a copy of table 23 on page 135 from [**44**]:

$$\begin{aligned}
PAL_0(w) &= s_1(\mathit{true}) \cdot PAL_0(w) && \big(|w| \geq 0\big) \\
PAL_{k+1}(\varepsilon) &= s_{k+2}(\mathit{true}) \cdot PAL_{k+1}(\varepsilon) \\
&\quad + \sum_{x \in S} r_{k+2}(x) \cdot s_{k+2}(\mathit{true}) \cdot PAL_{k+1}(x) \\
PAL_{k+1}(w) &= \sum_{x \in S} r_{k+2}(x) \cdot s_{k+2}\big(\mathit{ispal}(x \cdot w)\big) \\
&\qquad \cdot PAL_{k+1}(x \cdot w) && \big(0 < |w| < 2(k+1)\big) \\
PAL_{k+1}(w) &= \delta && \big(2(k+1) \leq |w|\big)
\end{aligned}$$

Where S is a finite set of symbols from which the input strings are built up. Now $|w|$ stands for the length of the string w. A predicate *ispal* is also used, with strings of symbols as its domain which is *true* iff its argument is a palindrome. Furthermore we have $B := \{true, false\}$. Now we will give hereinafter the specification of a cell C_i as it appears in [**44**]:

$$\begin{aligned}
C_i &= s_{i+1}(true) \cdot C_i + \sum_{x \in S} r_{i+1}(x) \cdot s_{i+1}(true) \cdot C_i'(x) \\
C_i'(x) &= \sum_{y \in S} r_{i+1}(y) \cdot \sum_{v \in B} r_i(v) \cdot C_i''(x, y, v) \\
&\quad + \sum_{v \in B} r_i(v) \cdot \sum_{y \in S} r_{i+1}(y) \cdot C_i''(x, y, v) \\
C_i''(x, y, v) &= \big(s_{i+1}\big((x = y) \mathbin{\&} v\big) \parallel s_i(y)\big) \cdot C_i'(x) \\
TC &= s_1(true) \cdot TC.
\end{aligned}$$

For the sake of clarity, we here eliminated the merge in the second equation, in order to obtain directly the correct form of the specification. Consider the case $i = 1$. If we compare the defining equations of $C_1'(x)$ with equations (14) and (15) of section 2.3, we can conclude with the use of RSP, that $P'' = C_1'(x)$. This means that we have, because of theorem (2.3.3):

$$\forall d, e \in D: \quad [d, e](X'') = C_1'(x),$$

where X'' is defined in equations (17) and (18) of section 2.3. What we want to do now, is to modify the specification of C_i by incorporating, in a notationally correct way, the defining equations of X'' in the whole specification. Now let us consider the result (where f_x is the function that we defined in equation (16) of section 2.3):

$$\begin{aligned}
X_i &= s_{i+1}(true) \cdot X_i + \sum_{x \in S} r_{i+1}(x) \cdot s_{i+1}(true) \cdot X_i(x) \\
X_i(x) &= \Big(\sum_{y \in S} r_{i+1}(y \downarrow 2i) \parallel \sum_{v \in B} r_i(v \downarrow 2i - 1)\Big) \cdot Y_i(x) \\
Y_i(x) &= \big(s_{i+1}\big(f_x(\uparrow 2i - 1, \uparrow 2i)\big) \parallel s_i\big(id(\uparrow 2i)\big)\big) \cdot X_i(x) \\
TC &= s_1(true) \cdot TC.
\end{aligned}$$

We have just seen that $[d, e](X'') = [d, e]\big(X_1(x)\big) = C_1'(x)$, and we also want, of course, that the following holds:

$$\forall d_1, \ldots, d_{2i} \in D: \quad [d_1, \ldots, d_{2i}]\big(X_i(x)\big) = C_i'(x).$$

This we can obtain by direct calculation, as was done in section 2.3 for the case $i = 1$, but we will prove this with the use of proposition (2.5.14). For, there is a bijection $\sigma : reg\big(X_1(x)\big) \longrightarrow \mathrm{reg}\big(X_i(x)\big)$ defined by the following two equations: $\sigma(1) = 2i-1$ and $\sigma(2) = 2i$. Let ρ be the renaming that transforms $C_1'(x)$ into $C_i'(x)$ (in fact $r_1 \mapsto r_i,\ r_2 \mapsto r_{i+1},\ s_1 \mapsto s_i,\ s_2 \mapsto s_{i+1}$). Now we can derive:

$$
\begin{aligned}
C_i'(x) &= \rho\big(C_1'(x)\big) \\
&= [d,e] \circ \rho\big(X_1(x)\big) && (2.3.3) \\
&= [d_1,\ldots,d_{2i-2},d,e] \circ \rho_{h_\sigma} \circ \rho\big(X_1(x)\big) && (2.5.14) \\
&= [d_1,\ldots,d_{2i-2},d,e]\big(X_i(x)\big).
\end{aligned}
$$

Subsequently we want to characterize the systolic array from which we will prove that it is bisimilar with $PAL_k(\varepsilon)$.

The encapsulation set H_k of actions resulting in a deadlock is defined as

$$H_k = \big\{\, s_i(x), r_i(y) \;\big|\; x,y \in S \quad \text{and } i < k+1 \big\},$$

and the abstraction set I of internal communication actions is defined as

$$I = \big\{\, c_i(x) \;\big|\; x \in S \quad \text{and } i \in \mathbb{N} \big\}.$$

Define

$$M(k) = \begin{cases} \tau_I \circ \partial_{H_k} \circ [r^{(k)}]\big(X_k \parallel \cdots \parallel X_1 \parallel TC\big), & \text{if } k > 0; \\ TC, & \text{if } k = 0 \end{cases}$$

where

$$[r^{(k)}] \in R_k = \big\{\, [d_1,\ldots,d_{2k}] \;\big|\; d_i \in D \ (:= S \cup B) \quad \text{for all } 1 \le i \le 2k \,\big\}.$$

Now we will show that the modified $M(k)$ is in fact the same as the $M(k)$ that is defined in [**44**]. To avoid ambiguity, we write $M'(k)$ for the one that is defined in [**44**]:

$$M'(k) = \mathrm{td}k(C_k \parallel \cdots \parallel C_1 \parallel TC).$$

Observe that we have for each $k > 0 : \mathrm{reg}(X_i) \cap \mathrm{reg}(X_j) = \emptyset$, for all $i, j \in \mathbb{N}$ with $1 \le i < j \le k$. This means that, because of theorem (2.5.11), the register operator distributes over the merge. This yields:

$$M(k) = \mathrm{td}k\big([r^{(k)}](X_k) \parallel \cdots \parallel [r^{(k)}](X_1) \parallel TC\big).$$

We can now evaluate $[r^{(k)}](X_i)$ for all $1 \le i \le k$:

$$[r^{(k)}](X_i) = s_{i+1}(true)[r^{(k)}](X_i) + \sum_{x \in S} r_{i+1}(x) \cdot s_{i+1}(true)[r^{(k)}]\big(X_i(x)\big).$$

Above, we have just seen the following: $[r^{(k)}]\big(X_i(x)\big) = C_i'(x)$. Observe that we have $2k$ registers instead of $2i$. In this case we may also use proposition (2.5.14), quod vide. Hence we have $[r^{(k)}](X_i) = C_i$. Thus we find $M'(k) = M(k)$.

Lemma (2.6.1)

Let $[r^{(k)}] = [d_1, \dots, d_{2k-2}, ispal(v), y] \in R_k$. Then, $\forall x, y \in S$, $\forall v \in S^*$ such that $|v| \leq 2k-1$, we have:

$$\begin{aligned}&\tau_I \circ \partial_{H_k} \circ [r^{(k)}]\big(Y_k(x) \parallel PAL_{k-1}(v)\big)\\ &\quad = \begin{cases} \tau \cdot s_{k+1}\big(ispal(y \cdot v \cdot x)\big) \cdot PAL_k(y \cdot v \cdot x), & \text{if } |v| < 2k-2;\\ s_{k+1}\big(ispal(y \cdot v \cdot x)\big) \cdot PAL_k(y \cdot v \cdot x), & \text{if } |v| = 2k-1.\end{cases}\end{aligned}$$

Proof. Let $[s^{(k)}] = [d_1, \dots, d_{2k-2}, w, z] \in R_k$. We claim that the following holds: $\forall x, y \in S$, $\forall v \in S^*$ with $|v| \leq 2k-1$:

$$\tau_I \circ \partial_{H_k} \circ [s^{(k)}]\big(X_k(x) \parallel s_k\big(ispal(v)\big) \cdot PAL_{k-1}(v)\big) = \tau \cdot PAL_k(v \cdot x).$$

Define

$$\begin{aligned}&Q_{(w,z)}(v, x)\\ &\quad = \begin{cases} \tau_I \circ \partial_{H_k} \circ [s^{(k)}]\big(X_k(x) \parallel s_k\big(ispal(v)\big) \cdot PAL_{k-1}(v)\big), & \text{if } |v| \leq 2k-1;\\ \delta, & \text{otherwise.}\end{cases}\end{aligned}$$

We use the following abbreviations:

$$\begin{aligned}[p^{(k)}] &= [d_1, \dots, d_{2k-2}, w, y]\\ [q^{(k)}] &= [d_1, \dots, d_{2k-2}, ispal(v), z]\end{aligned}$$

First we work out $Q_{(w,z)}(v, x)$ for the case $|v| \leq 2k-1$:

$$\begin{aligned}&Q_{(w,z)}(v, x)\\ &\quad = \tau_I \circ \partial_{H_k} \circ [s^{(k)}]\Bigg(\Big(\Big(\sum_{y \in S} r_{k+1}(y \downarrow 2k) \parallel \sum_{u \in B} r_k(u \downarrow 2k-1)\Big)\\ &\qquad \cdot Y_k(x) \parallel s_k\big(ispal(v)\big) \cdot PAL_{k-1}(v)\Bigg)\\ &\quad = \tau_I \circ \partial_{H_k} \circ [s^{(k)}]\Big(\sum_{y \in S} r_{k+1}(y \downarrow 2k)\\ &\qquad \cdot \Big(\sum_{u \in B} r_k(u \downarrow 2k-1) Y_k(x) \parallel s_k\big(ispal(v)\big) PAL_{k-1}(v)\Big)\\ &\qquad + c_k\big(ispal(v) \downarrow 2k-1\big) \cdot \sum_{z \in S} r_{k+1}(z \downarrow 2k) Y_k(x) \parallel PAL_{k-1}(v)\Big)\\ &\quad = \sum_{y \in S} r_{k+1}(y) \tau_I \circ \partial_{H_k} \circ [p^{(k)}]\Big(c_k\big(ispal(v) \downarrow 2k-1\big) \cdot \big(Y_k(x) \parallel PAL_{k-1}(v)\big)\\ &\qquad + \tau \cdot \tau_I \circ \partial_{H_k} \circ [q^{(k)}]\Big(\sum_{z \in S} r_{k+1}(z \downarrow 2k) \cdot \big(Y_k(x) \parallel PAL_{k-1}(v)\big)\end{aligned}$$

$$
\begin{aligned}
&= \sum_{y \in S} r_{k+1}(y) \cdot \tau \cdot \tau_I \circ \partial_{H_k} \circ [r^{(k)}]\big(Y_k(x) \parallel PAL_{k-1}(v)\big) \\
&\quad + \tau \cdot \sum_{y \in S} r_{k+1}(y) \tau_I \circ \partial_{H_k} \circ [r^{(k)}]\big(Y_k(x) \parallel PAL_{k-1}(v)\big) \\
&= \tau \cdot \sum_{y \in S} r_{k+1}(y) \tau_I \circ \partial_{H_k} \circ [r^{(k)}]\big(Y_k(x) \parallel PAL_{k-1}(v)\big).
\end{aligned}
$$

We will distinguish two subcases: $|v| < 2k-1$:

$$
\begin{aligned}
&\tau_I \circ \partial_{H_k} \circ [r^{(k)}]\big(Y_k(x) \parallel PAL_{k-1}(v)\big) \\
&\quad = \tau_I \circ \partial_{H_k} \circ [r^{(k)}]\Big(\big(s_{k+1}\big(f_x(\uparrow 2k-1, \uparrow 2k)\big) \parallel s_k\big(id(\uparrow 2k)\big)\big) \\
&\qquad \cdot X_k(x) \parallel PAL_{k-1}(v)\Big) \\
&\quad = \tau_I \circ \partial_{H_k} \circ [r^{(k)}]\Bigg(s_{k+1}\big(f_x(\uparrow 2k-1, \uparrow 2k)\big) \cdot \big(s_k\big(id(\uparrow 2k)\big) \\
&\qquad \cdot X_k(x) \parallel PAL_{k-1}(v)\big) \\
&\qquad + s_k\big(id(\uparrow 2k)\big) \cdot \big(s_{k+1}\big(f_x(\uparrow 2k-1, \uparrow 2k)\big) \cdot X_k(x) \parallel PAL_{k-1}(v)\big) \\
&\qquad + \sum_{x \in S} r_k(x)\big(s_k\big(ispal(x \cdot v)\big) \cdot PAL_k(x \cdot v) \parallel Y_k(x)\big) \\
&\qquad + \sum_{z \in S} c_k\big(z \wedge id(\uparrow 2k)\big) \cdot \big(s_k\big(ispal(z \cdot v)\big) \cdot PAL_{k-1}(z \cdot v) \\
&\qquad \parallel s_{k+1}\big(f_x(\uparrow 2k-1, \uparrow 2k)\big) \cdot X_k(x)\big)\Bigg) \\
&\quad = s_{k+1}\big((x=y) \;\&\; ispal(v)\big) \tau_I \circ \partial_{H_k} \circ [r^{(k)}]\big(s_k\big(id(\uparrow 2k)\big) \\
&\qquad \cdot X_k(x) \parallel PAL_{k-1}(v)\big) \\
&\qquad + \delta + \tau \cdot \tau_I \circ \partial_{H_k} \circ [r^{(k)}]\big(s_{k+1}\big(f_x(\uparrow 2k-1, \uparrow 2k)\big) \\
&\qquad \cdot X_k(x) \parallel s_k\big(ispal(y \cdot v)\big) \cdot PAL_{k-1}(y \cdot v)\big) \\
&\quad = s_{k+1}\big((x=y) \;\&\; ispal(v)\big) \cdot \tau_I \circ \partial_{H_k} \circ [r^{(k)}]\Big(\sum_{z \in S} c_k\big(z \wedge id(\uparrow 2k)\big) \\
&\qquad \cdot \big(X_k(x) \parallel s_k\big(ispal(z \cdot v)\big) \cdot PAL_{k-1}(z \cdot v)\big)\Big) \\
&\qquad + \tau \cdot s_{k+1}\big((x=y) \;\&\; ispal(v)\big) \\
&\qquad \cdot \tau_I \circ \partial_{H_k} \circ [r^{(k)}]\big(X_k(x) \parallel s_k\big(ispal(y \cdot v)\big) \cdot PAL_{k-1}(y \cdot v)\big) \\
&\quad = s_{k+1}\big((x=y) \;\&\; ispal(v)\big) \\
&\qquad \cdot \tau \cdot \tau_I \circ \partial_{H_k} \circ [r^{(k)}]\big(X_k(x) \parallel s_k\big(ispal(y \cdot v)\big) \cdot PAL_{k-1}(y \cdot v)\big) \\
&\qquad + \tau \cdot s_{k+1}\big((x=y) \;\&\; ispal(v)\big) \\
&\qquad \cdot \tau_I \circ \partial_{H_k} \circ [r^{(k)}]\big(X_k(x) \parallel s_k\big(ispal(y \cdot v)\big) \cdot PAL_{k-1}(y \cdot v)\big)
\end{aligned}
$$

$$\begin{aligned}
&= \tau \cdot s_{k+1}\big((x = y) \mathbin{\&} \mathit{ispal}(v)\big)\\
&\quad \cdot \tau_I \circ \partial_{H_k} \circ [r^{(k)}]\big(X_k(x) \parallel s_k\big(\mathit{ispal}(y \cdot v)\big) \cdot \mathit{PAL}_{k-1}(y \cdot v)\big)\\
&= \tau \cdot s_{k+1}\big((x = y) \mathbin{\&} \mathit{ispal}(v)\big) \cdot Q_{(\mathit{ispal}(v),y)}(y \cdot v, x),
\end{aligned}$$

since $|y \cdot v| \leq 2k-1$, if $|v| < 2k-1$. Now we consider the case: $|v| = 2k-1$.

$$\begin{aligned}
&\tau_I \circ \partial_{H_k} \circ [r^{(k)}]\big(Y_k(x) \parallel \mathit{PAL}_{k-1}(v)\big)\\
&\qquad = \tau_I \circ \partial_{H_k} \circ [r^{(k)}]\big(Y_k(x) \parallel \delta\big)\\
&\qquad = \tau_I \circ \partial_{H_k} \circ [r^{(k)}]\big(Y_k(x) \cdot \delta\big)\\
&\qquad = \tau_I \circ \partial_{H_k} \circ [r^{(k)}]\big(s_{k+1}\big(f_x(\uparrow 2k-1, \uparrow 2k)\big)\\
&\qquad\qquad \cdot s_k\big(\mathit{id}(\uparrow 2k)\big) \cdot Y_k(x) \cdot \delta\big) + \delta\\
&\qquad = s_{k+1}\big((x = y) \mathbin{\&} \mathit{ispal}(v)\big) \cdot \delta\\
&\qquad = s_{k+1}\big((x = y) \mathbin{\&} \mathit{ispal}(v)\big) \cdot Q_{(\mathit{ispal}(v),y)}(y \cdot v, x).
\end{aligned}$$

Note that we have:

$$\forall v \in S^*,\ |v| \leq 2k-1: \qquad (x = y) \mathbin{\&} \mathit{ispal}(v) \iff \mathit{ispal}(y \cdot v \cdot x),$$

so we find:

$$\begin{aligned}
&\tau_I \circ \partial_{H_k} \circ [r^{(k)}]\big(Y_k(x) \parallel \mathit{PAL}_{k-1}(v)\big)\\
&\qquad = \begin{cases} \tau \cdot s_{k+1}\big(\mathit{ispal}(yvx)\big) \cdot Q_{(\mathit{ispal}(v),y)}(yv, x), & \text{if } |v| < 2k-1;\\ s_{k+1}\big(\mathit{ispal}(yvx)\big) \cdot Q_{(\mathit{ispal}(v),y)}(yv, x), & \text{if } |v| = 2k-1. \end{cases}
\end{aligned}$$

After substitution we obtain for all $v \in S^*$ with $|v| \leq 2k-1$:

$$Q_{(w,z)}(v, x) = \tau \cdot \sum_{y \in S} r_{k+1}(y) \cdot s_{k+1}\big(\mathit{ispal}(yvx)\big) \cdot Q_{(\mathit{ispal}(v),y)}(yv, x).$$

Now consider the guarded recursive specification E below:

$$\begin{aligned}
E = \Big\{ T_{(w,z)}(v, x) = \tau \cdot \sum_{y \in S} r_{k+1}(y) \cdot s_{k+1}\big(\mathit{ispal}(yvx)\big) \cdot T_{(\mathit{ispal}(v),y)}(yv, x)\\
\Big|\ v \in S^*, |v| \leq 2k-1, w \in B, z, x \in S \Big\}
\end{aligned}$$

We immediately see that $Q_{(w,z)}(v, x)$ is a solution for E. Now consider the following:

$$\begin{aligned}
&\tau \cdot \sum_{y \in S} r_{k+1}(y) \cdot s_{k+1}\big(\mathit{ispal}(yvx)\big) \cdot \tau \cdot \mathit{PAL}_k(yvx)\\
&\qquad = \tau \cdot \sum_{y \in S} r_{k+1}(y) \cdot s_{k+1}\big(\mathit{ispal}(yvx)\big) \cdot \mathit{PAL}_k(yvx)\\
&\qquad = \tau \cdot \mathit{PAL}_k(vx).
\end{aligned}$$

Thus we see, if we put

$$T_{(w,z)}(v,x) = \tau \cdot PAL_k(vx)$$

then

$$T_{(ispal(v),y)}(yv,x) = \tau \cdot PAL_k(yvx).$$

This solves the system E. By RSP we find that $Q_{(w,z)}(v,x) = \tau \cdot PAL_k(vx)$ for all $v \in S^*$, $|v| \leq 2k-1$. This concludes the proof of the claim. Observe that we herewith implicitly proved 2.6.1, too.

Theorem (2.6.2)

$$\forall k \geq 0 : M(k) = PAL_k(\varepsilon).$$

Proof. If $k = 0$ there is nothing to prove. So let us assume henceforth that $k > 0$. We will first prove the following:

$$\forall k > 0 : \tau_I \circ \partial_{H_k} \circ [r^{(k)}]\big(X_k \parallel PAL_{k-1}(\varepsilon)\big) = PAL_k(\varepsilon). \tag{1}$$

Below we will compute, for the left-hand side of (1), two more equations:

$$\begin{aligned}
&\tau_I \circ \partial_{H_k} \circ [r^{(k)}]\big(X_k \parallel PAL_{k-1}(\varepsilon)\big)\\
&\qquad = \tau_I \circ \partial_{H_k} \circ [r^{(k)}]\big(X_k \mathbin{\lfloor\!\lfloor} PAL_{k-1}(\varepsilon)\big)\\
&\qquad = s_{k+1}(true) \cdot \tau_I \circ \partial_{H_k} \circ [r^{(k)}]\big(X_k \parallel PAL_{k-1}(\varepsilon)\big)\\
&\qquad\quad + \sum_{x \in S} r_{k+1}(x) \cdot \tau_I \circ \partial_{H_k} \circ [r^{(k)}]\big(s_{k+1}(true)\\
&\qquad\qquad \cdot X_k(x) \parallel PAL_{k-1}(\varepsilon)\big)\\
&\qquad = s_{k+1}(true) \cdot \tau_I \circ \partial_{H_k} \circ [r^{(k)}]\big(X_k \parallel PAL_{k-1}(\varepsilon)\big)\\
&\qquad\quad + \sum_{x \in S} r_{k+1}(x) \cdot \tau_I \circ \partial_{H_k} \circ [r^{(k)}]\Big(s_{k+1}(true)\\
&\qquad\qquad \cdot \big(X_k(x) \parallel PAL_{k-1}(\varepsilon)\big)\Big)\\
&\qquad = s_{k+1}(true) \cdot \tau_I \circ \partial_{H_k} \circ [r^{(k)}]\big(X_k \parallel PAL_{k-1}(\varepsilon)\big)\\
&\qquad\quad + \sum_{x \in S} r_{k+1}(x) \cdot s_{k+1}(true)\\
&\qquad\qquad \cdot \tau_I \circ \partial_{H_k} \circ [r^{(k)}]\big(X_k(x) \parallel PAL_{k-1}(\varepsilon)\big),
\end{aligned} \tag{2}$$

and

$$\begin{aligned}
&\tau_I \circ \partial_{H_k} \circ [r^{(k)}]\big(X_k(x) \parallel PAL_{k-1}(\varepsilon)\big)\\
&\quad = \tau_I \circ \partial_{H_k} \circ [r^{(k)}]\big(X_k(x) \mathbin{\lfloor\!\lfloor} PAL_{k-1}(\varepsilon) + X_k(x) \mid PAL_{k-1}(\varepsilon)\big)
\end{aligned}$$

$$\begin{aligned}
&= \tau_I \circ \partial_{H_k} \circ [r^{(k)}]\Big(\sum_{y\in S} r_{k+1}(y \downarrow 2k)\\
&\quad \cdot \Big(\sum_{v\in B} r_k(v \downarrow 2k-1)\cdot Y_k(x) \parallel PAL_{k-1}(\varepsilon)\Big)\\
&\quad + \sum_{w\in B} r_k(w \downarrow 2k-1)\cdot \Big(\sum_{z\in S} r_{k+1}(z \downarrow 2k)\cdot Y_k(x) \parallel PAL_{k-1}(\varepsilon)\Big)\\
&\quad + c_k(true \downarrow 2k-1)\cdot \Big(\sum_{z\in S} r_{k+1}(z \downarrow 2k)\cdot Y_k(x) \parallel PAL_{k-1}(\varepsilon)\Big)\Big)\\
&= \sum_{y\in S} r_{k+1}(y)\tau_I \circ \partial_{H_k} \circ [p^{(k)}]\Big(\sum_{v\in B} r_k(v \downarrow 2k-1)\cdot Y_k(x) \parallel PAL_{k-1}(\varepsilon)\Big)\\
&\quad + \delta + \tau\cdot\tau_I \circ \partial_{H_k} \circ [q^{(k)}]\Big(\sum_{z\in S} r_{k+1}(z \downarrow 2k)\cdot Y_k(x) \parallel PAL_{k-1}(\varepsilon)\Big)\\
&= \sum_{y\in S} r_{k+1}(y)\tau_I \circ \partial_{H_k} \circ [p^{(k)}]\Big(\sum_{v\in B} r_k(v \downarrow 2k-1)\cdot Y_k(x) \mid PAL_{k-1}(\varepsilon)\Big)\\
&\quad + \tau\cdot\tau_I \circ \partial_{H_k} \circ [q^{(k)}]\Big(\sum_{z\in S} r_{k+1}(z \downarrow 2k)\cdot Y_k(x) \mathbin{\|\!\underline{\ \ }} PAL_{k-1}(\varepsilon)\Big)\\
&= \sum_{y\in S} r_{k+1}(y)\cdot\tau_I \circ \partial_{H_k} \circ [p^{(k)}]\Big(c_k(true \downarrow 2k-1)\cdot \big(Y_k(x) \parallel PAL_{k-1}(\varepsilon)\big)\Big)\\
&\quad + \tau\cdot \sum_{y\in S} r_{k+1}(y)\cdot\tau_I \circ \partial_{H_k} \circ [s^{(k)}]\big(Y_k(x) \parallel PAL_{k-1}(\varepsilon)\big) \qquad (3)\\
&= \sum_{y\in S} r_{k+1}(y)\cdot\tau\cdot\tau_I \circ \partial_{H_k} \circ [s^{(k)}]\big(Y_k(x) \parallel PAL_{k-1}(\varepsilon)\big)\\
&\quad + \tau\cdot \sum_{y\in S} r_{k+1}(y)\cdot\tau_I \circ \partial_{H_k} \circ [s^{(k)}]\big(Y_k(x) \parallel PAL_{k-1}(\varepsilon)\big)\\
&= \tau\cdot \sum_{y\in S} r_{k+1}(y)\cdot\tau_I \circ \partial_{H_k} \circ [s^{(k)}]\big(Y_k(x) \parallel PAL_{k-1}(\varepsilon)\big) \qquad (4)
\end{aligned}$$

Where for notational purpose we renamed in (3) the variable z. Furthermore we used the abbreviations:

$$\begin{aligned}
[r^{(k)}] &= [d_1,\ldots,d_{2k}],\\
[p^{(k)}] &= [d_1,\ldots,d_{2k-1},y],\\
[q^{(k)}] &= [d_1,\ldots,d_{2k-2},true,d_{2k}],\\
[s^{(k)}] &= [d_1,\ldots,d_{2k-2},true,y].
\end{aligned}$$

Recall that we have:

$$[s^{(k)}] = [d_1,\ldots,d_{2k-2},true,y] = [d_1,\ldots,d_{2k-2},ispal(\varepsilon),y].$$

So with lemma (2.6.1) and equation (4) we conclude

$$\begin{aligned}
&\tau_I \circ \partial_{H_k} \circ [r^{(k)}]\big(X_k(x) \parallel PAL_{k-1}(\varepsilon)\big) \\
&\quad = \tau \cdot \sum_{y \in S} r_{k+1}(y) \tau_I \circ \partial_{H_k} \circ [s^{(k)}]\big(Y_k(x) \parallel PAL_{k-1}(\varepsilon)\big) && (4) \\
&\quad = \tau \cdot \sum_{y \in S} r_{k+1}(y) s_{k+1}\big(ispal(y \cdot \varepsilon \cdot x)\big) PAL_k(y \cdot \varepsilon \cdot x) && (2.6.1) \\
&\quad = \tau \cdot \sum_{y \in S} r_{k+1}(y) s_{k+1}\big(ispal(yx)\big) PAL_k(yx) \\
&\quad = \tau \cdot PAL_k(x).
\end{aligned}$$

This we may substitute in equation (2), and so we achieve:

$$\begin{aligned}
&\tau_I \circ \partial_{H_k} \circ [r^{(k)}]\big(X_k \parallel PAL_{k-1}(\varepsilon)\big) \\
&\quad = s_{k+1}(true) \tau_I \circ \partial_{H_k} \circ [r^{(k)}]\big(X_k \parallel PAL_{k-1}(\varepsilon)\big) \\
&\qquad + \sum_{x \in S} r_{k+1}(x) \cdot s_{k+1}(true) \cdot \tau_I \circ \partial_{H_k} \circ [r^{(k)}]\big(X_k(x) \parallel PAL_{k-1}(\varepsilon)\big) \\
&\quad = s_{k+1}(true) \tau_I \circ \partial_{H_k} \circ [r^{(k)}]\big(X_k \parallel PAL_{k-1}(\varepsilon)\big) \\
&\qquad + \sum_{x \in S} r_{k+1}(x) \cdot s_{k+1}(true) \cdot PAL_k(x).
\end{aligned}$$

Consider the following guarded recursive specification E:

$$\begin{aligned}
E = \Big\{\; & Z_k = s_{k+1}(true) \cdot Z_k + \sum_{x \in S} r_{k+1}(x) \cdot s_{k+1}(true) \cdot Z_k(x) \\
& Z_k(w) = \begin{cases} \sum_{x \in S} r_{k+1}(x) \cdot s_{k+1}\big(ispal(x \cdot w)\big) \cdot Z_k(x \cdot w), & \text{if } 0 < |w| < 2k; \\ \delta, & \text{otherwise.} \end{cases} \\
& \;\big|\; w \in S^*, k > 0 \Big\}
\end{aligned}$$

If we put $Z_k = PAL_k(\varepsilon)$ and $Z_k(w) = PAL_k(w)$, then it is clear from the defining equations of PAL_k, that $PAL_k(\varepsilon)$ is a solution for E. If we put $Z_k = \tau_I \circ \partial_{H_k} \circ [r^{(k)}]\big(X_k \parallel PAL_{k-1}(\varepsilon)\big)$ and $Z_k(w) = PAL_k(w)$, it is evident from the hereinbefore inferred equation, that this system also is a solution for E. Hence, with RSP we conclude:

$$\forall k > 0 : \tau_I \circ \partial_{H_k} \circ [r^{(k)}]\big(X_k \parallel PAL_{k-1}(\varepsilon)\big) = PAL_k(\varepsilon).$$

This finishes the proof of claim (1). Now we find:

$$\begin{aligned}
PAL_k(\varepsilon) &= \tau_I \circ \partial_{H_k} \circ [r^{(k)}]\big(X_k \parallel PAL_{k-1}(\varepsilon)\big) \\
&= \mathrm{td}k\big([r^{(k)}](X_k) \parallel PAL_{k-1}(\varepsilon)\big)
\end{aligned}$$

$$
\begin{aligned}
&= \mathrm{td}k\bigl([r^{(k)}](X_k) \parallel \bigl(\tau_I \circ \partial_{H_{k-1}} \circ [r^{(k-1)}]\bigl(X_{k-1} \parallel PAL_{k-2}(\varepsilon)\bigr)\bigr)\\
&= \mathrm{td}k\bigl([r^{(k)}](X_k) \parallel \bigl(\mathrm{td}k-1\bigl([r^{(k)}]X_{k-1} \parallel PAL_{k-2}(\varepsilon)\bigr)\bigr)\\
&\vdots\\
&= \mathrm{td}k\bigl([r^{(k)}](X_k) \parallel \bigl(\mathrm{td}k-1\bigl([r^{(k)}](X_{k-1})\\
&\qquad \parallel \cdots \mathrm{td}1\bigl([r^{(k)}](X_1) \parallel PAL_0(\varepsilon)\bigr)\cdots\bigr)\bigr)\\
&= \mathrm{td}k\bigl([r^{(k)}](X_k) \parallel \bigl(\mathrm{td}k-1\bigl([r^{(k)}](X_{k-1})\\
&\qquad \parallel \cdots \mathrm{td}1\bigl([r^{(k)}](X_1) \parallel TC\bigr)\cdots\bigr)\bigr)\\
&= \mathrm{td}k\bigl(C_k \parallel \bigl(\mathrm{td}k-1\bigl(C_{k-1}\bigr) \parallel \cdots \mathrm{td}1\bigl(C_1\bigr) \parallel TC\bigr)\cdots\bigr)\bigr).
\end{aligned}
$$

At this point we return to the original proof, so hereinafter we will copy out the last piece. In this last piece of the proof we will make use of the so-called conditional alphabet axioms. For the definition of an alphabet and the conditional axioms we refer to section 1.5. It is easy to prove by induction, that:

$$
\begin{aligned}
\alpha(C_k) \bigm| \bigl(\alpha\bigl(M(k-1)\bigr) \cap H_k\bigr) &= \emptyset,\\
\alpha(C_k) \bigm| \bigl(\alpha\bigl(M(k-1)\bigr) \cap I\bigr) &= \emptyset.
\end{aligned}
$$

So because of $H_{k-1} \subseteq H_k$ we find, using the conditional axioms CA1 and CA2 of table 1.7 (page 19):

$$
\begin{aligned}
&\mathrm{td}k\bigl(\mathrm{td}k-1\bigl(\cdots\bigl(\mathrm{td}1(C_1 \parallel TC)\bigr)\cdots\bigr)\bigr)\\
&\quad = \tau_I \circ \cdots \circ \mathrm{td}k \circ \cdots \circ \partial_{H_1}\bigl(C_k \parallel \cdots \parallel C_1 \parallel TC\bigr)\\
&\quad = \mathrm{td}k\bigl(C_k \parallel \cdots \parallel C_1 \parallel TC\bigr)\\
&\quad = \tau_I \circ \partial_{H_k} \circ [r^{(k)}](X_k \parallel \cdots \parallel X_1 \parallel TC)\\
&\quad = M(k).
\end{aligned}
$$

This proves theorem 2.6.2.

Remark (2.6.3)

In the proof of theorem (2.6.2) we used axioms concerning abstraction. This axiom is known as the second τ-law of Milner (T2). To prove equation (1), however, we only used T2 in a context of the form

$$\bigl(\cdots a \cdot (\tau \cdot x + x)\cdots\bigr),$$

which can be proved to be an instance of H2. Where H2 is as follows:

$$a\bigl(\tau(y+z)+y\bigr) = a(y+z).$$

Thence, theorem (2.6.2) can be proved in *ACP* with branching bisimulation (see [**23**]).

Chapter 3

An Operator Definition Principle

IN a first attempt to bring some structure in the use of linear unary operators in process algebra, we propose two principles, in order to obtain a uniform approach for the introduction of these operators. For this we will use the notion of a linear functional specification that consists of functional equations and boundary conditions; the operator definition principle states that such a system has a solution. Furthermore, we define an operator specification principle; this principle states that a linear functional specification has at most one solution. Examples to demonstrate the usage of the two principles are given. We can think of the use of auxiliary operators in verifications, specifications, or a combination of both. Although these specific operators are, in general, not usable in another context, there is a need for auxiliary operators that can be defined as we wish. Moreover, we will need even more complicated auxiliary linear unary operators in the future, and therefore, a more sophisticated definition of a linear functional specification. So, suggestions to generalize this concept are given, too.

3.1. Introduction

We will consider in this chapter the algebra of communicating processes with abstraction and linear unary operators ($ACP_{\tau,u}$). We will introduce $ACP_{\tau,u}$ as an extension of ACP_{τ}, the algebra of communicating processes with abstraction, which was first studied in [**12**]. For a short introduction to ACP_{τ} we refer to chapter 1. Hereinafter, we will give an overview of what can be expected in the subsequent sections and we will give some motivation for the theory that will be presented in this chapter.

In section 3.2 we will give the signature and the axioms of $ACP_{\tau,u}$. Furthermore, we will formulate the operator definition principle and the operator specification principle in terms of solutions for linear functional specifications. In $ACP_{\tau,u}$, we will have two sorts: the sort of processes and the sort of linear unary operators. In the sort of processes we will have the "usual" constants: the atomic actions and the special constants. In the sort of linear unary operators the constants will be the projection operators, the encapsulation operator and the abstraction operator. It can be found that, with the aid of this theory, we can introduce the latter two operators. We still added them to the set of constants, since they are *not* auxiliary: with these operators we want to formulate other axioms. We can mention here $KFAR_n$; see definition (1.2.7) and conditional axioms; see section 1.5. The same yields for the projection operators: we formulate the approximation induction principle with them. So, even in a generalized version of this theory in which we can specify the projection operators (see section 3.8), we want to have these operators as constants. We will redefine certain items that are already known in ACP_τ, for instance, the notion of a guard and of a guarded recursive specification; see section 1.2. This will be done in definition (3.2.15). We will give a motivation on the alterations that we made to these notions. We will introduce two more principles: the recursive definition principle (*RDP*) and the recursive specification principle (*RSP*). These principles can be found in section 1.2. We will also introduce a more restrictive form of the approximation induction principle (AIP^-). We will repeat these notions, since they use the definition of a guarded recursive specification; and it is this definition that will be not the same as the one in chapter 1.

In section 3.3 we will prove the termination of the system that we introduced in section 3.2 by means of a method that is called the recursive path ordering. In [**12**] this method is used to prove the termination of a term rewrite system for ACP_τ. We will use the same method to prove the termination of a term rewrite system for $ACP_{\tau,u}$. We will explain this method and we will give all the necessary definitions. Thereinafter, we will prove an elimination result, which states that every closed $ACP_{\tau,u}$-term can be rewritten into a closed $BPA_{\delta,\tau}$-term; see table 3.3 on page 91.

In section 3.4 we will introduce some more concepts that are necessary in order to state general theorems concerning linear unary operators, which can be defined with the aid of linear functional specifications. In this section it will become clear that with this theory we do not longer have to prove for each auxiliary linear unary operator separately that it has certain properties: it will be sufficient to verify that this operator satisfies the conditions stated in these general theorems in order to know that it has the desired properties. We attempted to handle a great variety of subjects. But it will be far from complete. Many questions could be asked, and answered, on the subject of linear unary operators. For instance, we want to know what processes are fixed points of a linear unary operator, or: “When do linear unary operators commute?” Just to mention a few.

In section 3.5 we will construct a model for ACP_u. This is the axiom system $ACP_{\tau,u}$, but without abstraction. We did this because we wanted to construct what is called the standard model of process algebra, that is, the inverse or projective limit model. Moreover, it is a well-known fact that the combination of the projective limit model with the concept of abstraction is very problematic. We will prove that a desired property (such as *RDP*, or *OSP*) is valid for the elements of the inverse system and we use this to prove that it is preserved by taking the inverse limit. We think that some of these proofs can be shortened by using so-called preservation theorems on inverse limits, but since the theory on preservation theorems is only developed for single-sorted algebras, we will give direct proofs.

In section 3.6 we will apply the theory by giving some examples of verifications and specifications with the aid of auxiliary linear unary operators. We will also see that this theory is used to prove certain properties of operators that we already know. For instance, if we have the abstraction operator τ_I (this operator renames all the atomic actions that are in the subset $I \subseteq A$ into the silent step τ), then we immediately "feel" that $\tau_I \circ \tau_I$ must be τ_I. With the aid of *OSP* it is, actually, very trivial to prove this. It turns out that, with the aid of the general theorems in section 3.4, it will be sufficient to verify that τ_I^2 and τ_I are the same on the set of atomic actions in order to conclude that they are equal.

In section 3.7 we will give some suggestions how we can generalize this theory in order to be able to describe more (auxiliary) linear unary operators. Such as the projection operators or the generalized state operator, just to mention a few.

In section 3.8 we will emphasize that this theory can be the beginning of the unification of all the theories, built up from *ACP* or ACP_τ and, in addition, a number of auxiliary linear unary operators.

3.2. Definitions

In this section we will consider the algebra of communicating processes with abstraction and linear unary operators. We will use the following notation for this system: $ACP_{\tau,u}$. First, we will give a graphical representation of the signature of $ACP_{\tau,u}$ in figure 3.1. Subsequently, we will enumerate the same signature in a more textual way.

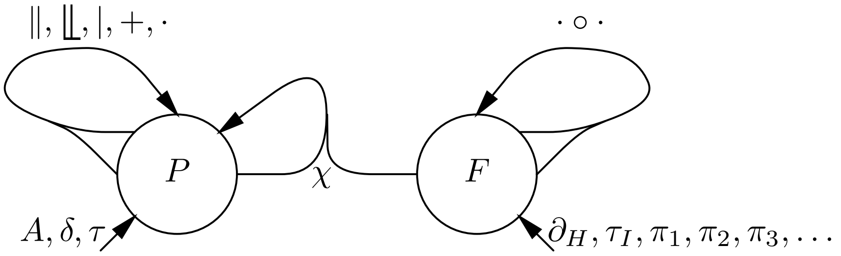

Figure 3.1. Graphical representation of the signature.

First, we consider the sort P of processes. We have a set of constants A, or atomic actions. Each $a \in A$ is a constant in P. We have also two special constants in P: δ or deadlock, and τ or silent step. Consider the binary operators (they are all infix operators):

merge:	$\| : P \times P \longrightarrow P,$
left-merge:	$\lfloor\!\lfloor : P \times P \longrightarrow P,$
communication-merge:	$\mid : P \times P \longrightarrow P,$
sequential composition:	$\cdot : P \times P \longrightarrow P,$
alternative composition:	$+ : P \times P \longrightarrow P.$

Now we consider the sort F of linear unary operators. For each $I \subseteq A$ we have a constant $\tau_I \in F$, which is called the abstraction operator. For each $H \subseteq A$ we have a constant ∂_H in F. This constant is called the encapsulation operator. For each $n \geq 1$ we have a constant $\pi_n \in F$. The constant π_n is called the (nth) projection operator. There is one binary operation with both arguments of sort F; it is the composition of functions: $\circ : F \times F \longrightarrow F$. Finally, we have a binary operation $\chi : F \times P \longrightarrow P$. This operator can be called the application operator. This concludes our discussion on the signature of $ACP_{\tau,u}$. Hereinafter, we will give the axiom system of $ACP_{\tau,u}$ in table 3.1 on page 73. In this table we use the following notational conventions: a, b, c are atomic actions or δ (we abbreviate $A_\delta = A \cup \{\delta\}$); x, y, z are processes; γ is a special constant (we use $C = \{\delta, \tau\}$ for the set of special constants); $n \geq 1$, and finally, f, g and h are linear unary operators.

At this point we will formulate an extensionality axiom which states: functions that behave the same on all processes are indeed the same. We will not use the axiom in this chapter but we added it since there is no reason not to have it.

Axiom (3.2.1) *Extensionality*

Let f and g be linear unary operators. If for all $x \in P : \chi(f, x) = \chi(g, x)$, then $f = g$. We will use the abbreviation EA for this axiom.

Definition (3.2.2)

Let N be a finite set of function names. A linear functional specification $E(N)$ for N is a set of the following form:

$$E(N) = \{\, r_{n,a} \mid n \in N, a \in A \,\} \cup \{\, e_{n,a} \mid n \in N, a \in A \,\}. \tag{1}$$

Both $r_{n,a}$ and $e_{n,a}$ are equations. Let $a \in A$ and $n \in N$ be fixed. Then we define the two equations $r_{n,a}$ and $e_{n,a}$ for this particular pair (n, a). The first equation $r_{n,a}$ is called a boundary condition and has the following form: there is an element $b \in A \cup C$ such that

$$r_{n,a} \equiv \chi(n, a) = b.$$

A1	$x + y = y + x$	$x \cdot \tau = x$	T1
A2	$x + (y + z) = (x + y) + z$	$\tau \cdot x + x = x$	T2
A3	$x + x = x$	$a(\tau \cdot x + y) = a \cdot (\tau \cdot x + y) + a \cdot x$	T3
A4	$(x + y) \cdot z = x \cdot z + y \cdot z$		
A5	$(x \cdot y) \cdot z = x \cdot (y \cdot z)$	$\tau \mathbin{\lfloor\!\lfloor} x = \tau \cdot x$	TM1
A6	$x + \delta = x$	$(\tau \cdot x) \mathbin{\lfloor\!\lfloor} y = \tau \cdot (x \parallel y)$	TM2
A7	$\delta \cdot x = \delta$	$\tau \mid x = \delta$	TC1
		$x \mid \tau = \delta$	TC2
C1	$a \mid b = b \mid a$	$(\tau \cdot x) \mid y = x \mid y$	TC3
C2	$(a \mid b) \mid c = a \mid (b \mid c)$	$x \mid (\tau \cdot y) = x \mid y$	TC4
C3	$\delta \mid a = \delta$		
		$\chi(\tau_I, a) = a, \quad \text{if } a \notin I$	TI1
CM1	$x \parallel y = x \mathbin{\lfloor\!\lfloor} y + y \mathbin{\lfloor\!\lfloor} x + x \mid y$	$\chi(\tau_I, a) = \tau, \quad \text{if } a \in I$	TI2
CM2	$a \mathbin{\lfloor\!\lfloor} x = a \cdot x$	$\chi(\tau_I, x \cdot y) = \chi(\tau_I, x) \cdot \chi(\tau_I, y)$	TI3
CM3	$(a \cdot x) \mathbin{\lfloor\!\lfloor} y = a \cdot (x \parallel y)$		
CM4	$(x + y) \mathbin{\lfloor\!\lfloor} z = x \mathbin{\lfloor\!\lfloor} z + y \mathbin{\lfloor\!\lfloor} z$	$\chi(f \circ g, x) = \chi\big(f, \chi(g, x)\big)$	XC1
CM5	$(a \cdot x) \mid b = (a \mid b) \cdot x$	$\chi\big((f \circ g) \circ h, x\big) = \chi\big(f \circ (g \circ h), x\big)$	XC2
CM6	$a \mid (b \cdot x) = (a \mid b) \cdot x$		
CM7	$(a \cdot x) \mid (b \cdot y) = (a \mid b) \cdot (x \parallel y)$	$\chi(f, \gamma) = \gamma$	X1
CM8	$(x + y) \mid z = x \mid z + y \mid z$	$\chi(f, \gamma \cdot x) = \gamma \cdot \chi(f, x)$	X2
CM9	$x \mid (y + z) = x \mid y + x \mid z$	$\chi(f, x + y) = \chi(f, x) + \chi(f, y)$	X3
D1	$\chi(\partial_H, a) = a, \quad \text{if } a \notin H$	$\chi(\pi_n, a) = a$	PR1
D2	$\chi(\partial_H, a) = \delta, \quad \text{if } a \in H$	$\chi(\pi_1, a \cdot x) = a$	PR2
D3	$\chi(\partial_H, x \cdot y) = \chi(\partial_H, x) \cdot \chi(\partial_H, y)$	$\chi(\pi_{n+1}, a \cdot x) = a \cdot \chi(\pi_n, x)$	PR3

Table 3.1. $ACP_{\tau,u}$.

The second equation $e_{n,a}$ is called a (linear) functional equation and it is of the following form:

$$e_{n,a} \equiv \chi(n, a \cdot x) = b \cdot \chi(m, x) \quad \text{for one } m \in N.$$

Examples of linear functional specifications can be found in remarks (3.2.11) and (3.2.19). Furthermore, if the set of function names contains only one element , say n, we will omit the braces in the notation of (1): we will write $E(n)$ instead of $E(\{n\})$.

Remark (3.2.3)

We will give a more explicit formulation of the definition of a linear functional specification. Let $N = \{n_1, \ldots, n_k\}$ be a set of function names. Let $\sigma : A \times \{1, \ldots, k\} \longrightarrow \{1, \ldots, k\}$ be a map. Now we define a linear functional specification to be the following:

$$\begin{aligned} E(N) = \big\{\, & n_i(a) = a^{(i)} : a \in A,\ 1 \leq i \leq k \,\big\} \\ & \cup \big\{\, n_i(a \cdot x) = a^{(i)} \cdot n_{\sigma(a,i)}(x) : a \in A,\ 1 \leq i \leq k \,\big\}, \end{aligned}$$

with $a^{(i)} \in A \cup \{\delta, \tau\}$. It will be clear that this formulation of a linear functional equation is equivalent to definition (3.2.2).

We will now formulate two principles which say something about the solutions for this sort of equational systems: the operator definition principle (*ODP*) and the operator specification principle (*OSP*). But first, we will need a definition to map the function names into the set of linear unary operators.

Definition (3.2.4)

We will call a mapping $\varphi : N \longrightarrow F$ a valuation for N.

Principle (3.2.5) *The Operator Definition Principle*

Let $E(N)$ be a linear functional specification for a set of function names N. Then the following holds: there is a valuation φ for N which solves the system of equations $E(N)$. We will use the compact notation *ODP* for this principle.

Principle (3.2.6) *The Operator Specification Principle*

Let $E(N)$ be a linear functional specification for a set of function names N. Then there is at most one valuation φ for N such that φ solves the system of equations $E(N)$. We will use the compact notation *OSP*.

Remarks (3.2.7)

See remarks (3.2.11) and (3.2.19) for examples of the use of both principles *ODP* and *OSP*. Henceforth, we will use for the expression $\chi(f, x)$, with $f \in F$ and $x \in P$, the more usual notation $f(x)$, provided that no confusion can arise. If n is a function name we will use this convention, too. If we have specified the boundary conditions of a certain function name n and we want this function name to respect the sequential composition, we will abbreviate this by stating the functional equation for n as follows:

$$n(a \cdot x) = n(a) \cdot n(x)$$

instead of reiterating the results of $n(a)$ for each $a \in A$. Examples of linear functional specifications in which a function name distributes over the sequential composition can be found in remark (3.2.11) and in section 3.6.

Definition (3.2.8)

Let $f \in F$ be a linear unary operator. We will define a subset $D(f) \subseteq F$ as the smallest set satisfying:

(i) $\quad f \in D(f)$

(ii) $\quad \big(g \in F,\, a \in A,\, b \in A \cup \{\tau\},\, \forall x \in P : f(a \cdot x) = b \cdot g(x)\big) \Longrightarrow g \in D(f)$

(iii) $\quad h \in D(g),\, g \in D(f) \Longrightarrow h \in D(f)$.

We will call $D(f)$ the set of derived operators of f.

Remarks (3.2.9)

We will calculate here for a number of linear unary operators, their sets of derived operators. We will start with π_3. Because of condition (i), we see that $\pi_3 \in D(\pi_3)$. We know that for all $a \in A, x \in P : \pi_3(a \cdot x) = a \cdot \pi_2(x)$. So we find with (ii) that $\pi_2 \in D(\pi_3)$. We also know that $\pi_2(a \cdot x) = a \cdot \pi_1(x)$ for all $a \in A$ and $x \in P$; thus we find with (ii) that $\pi_1 \in D(\pi_2)$. And with the aid of (iii) we see that $\pi_1 \in D(\pi_3)$. Observe that we do not have *constant* linear unary operators. For, let $c : P \longrightarrow P$ be as follows. For all $x \in P : c(x) = c$, with $c \in P$. Suppose that $c \in F$, then we know that $c(\delta) = \delta$, because of X1. So we find that $c = \delta$. But on the other hand we know that $c(\tau) = \tau$, so we see that $c = \tau$. Since $\delta \neq \tau$ we see that the assumption that $c \in F$ cannot hold. So in particular, we do not have in F a linear unary operator ℓ, with for all $x \in P : \ell(x) = \tau$. We know that $\pi_1(a \cdot x) = a = a \cdot \tau = a \cdot \ell(x)$. But $\ell \notin F$. So we find that $D(\pi_1) = \{\pi_1\}$. We thus obtain $D(\pi_3) = \{\pi_1, \pi_2, \pi_3\}$. It is very easy to deduce that $D(\pi_n) = \{\pi_1, \ldots, \pi_n\}$, for $n \geq 1$. The facts that $D(\tau_I) = \{\tau_I\}$ and $D(\partial_H) = \{\partial_H\}$ are also easy to verify. In remark (3.2.19) we have two linear unary operators μ and ν. It is immediately clear that $D(\mu) = D(\nu) = \{\mu, \nu\}$ q.v..

We left out δ in the conditions of definition (3.2.8)(ii). We did that since we otherwise would have $D(\partial_H) = F$ if $H \neq \emptyset$. For, we find for all $f \in F$:

$$\partial_H(h \cdot x) = \delta = \delta \cdot f(x).$$

This would imply that $f \in D(\partial_H)$, thus: $F = D(\partial_H)$. Without the δ we find $D(\partial_H) = \{\partial_H\}$. And this is exactly what it should be: the set of all the operators that ∂_H can become after "passing" the right atomic actions.

Definition (3.2.10)

Let $f \in F$ be a linear unary operator, which can be defined with the aid of a linear functional specification. If $|D(f)| = 1$ we will call f a (linear unary) renaming operator.

Remark (3.2.11)

We will repeat the definition of the renaming operator as it is stated in chapter 1, in order to make a comparison with the renaming operators that we defined hereinbefore in definition (3.2.10). Let $f : A \longrightarrow A \cup C$ be a function. Define an operator ρ_f as follows.

(i)	$\rho_f(\gamma) = \gamma$
(ii)	$\rho_f(a) = f(a)$
(iii)	$\rho_f(x \cdot y) = \rho_f(x) \cdot \rho_f(y)$
(iv)	$\rho_f(x + y) = \rho_f(x) + \rho_f(y)$

Consider the linear functional specification $E(r)$ for the singleton of function names $\{r\}$.

$$E(r) = \left\{ r(a) = f(a) : a \in A \right\} \\ \cup \left\{ r(a \cdot x) = r(a) \cdot r(x) : a \in A \right\}.$$

According to *ODP* there is a valuation $\varphi : \{r\} \longrightarrow F$ which solves the system of equations $E(r)$. Let us say $\varphi(r) = \rho$. It is evident that $|D(\rho)| = 1$; so we know, by definition, that ρ is a renaming operator. We also see that ρ_f is a solution for this system: $\rho_f(a) = f(a)$ and $\rho_f(a \cdot x) = \rho_f(a) \cdot \rho_f(x)$. But according to *OSP*, there is at most one solution, so thus we find: $\rho_f = \rho$. We have seen that the "old" renaming operators can be defined in terms of the "new" ones. In fact, these definitions are equivalent, since a linear functional specification that defines a renaming operator can only be of the form that $E(r)$ has.

Definition (3.2.12)

A linear unary operator $f \in F$ is called an abstracting operator if there is a linear unary operator $g \in D(f)$, such that $g(a) = \tau$, for some $a \in A$. Otherwise, $f \in F$ is called a concrete operator. Observe that the abstraction operator $\tau_I \in F$ is an abstracting operator (if $I \neq \emptyset$).

Example (3.2.13)

Let μ, ν be the linear unary operators considered in (3.2.19). We already saw in (3.2.9) that μ is an element of $D(\nu)$. In the linear functional specification, which defined both operators we can see that $\mu(i) = \tau$, so ν is an abstracting operator.

Subsequently, we will introduce two more principles: the recursive definition principle (*RDP*) and the recursive specification principle (*RSP*). (They can be found in chapter 1 but we will define their accompanying notions slightly differently.) For these principles, we need the notion of a (guarded) recursive specification. These notions are taken from [**16**], although the notion of a guard can already be found in [**14**] in process algebra. The definitions of such specifications are given hereinafter.

Definition (3.2.14)

Let X be a set of variables. A recursive specification E with variable set X over $ACP_{\tau,u}$ is a system of recursion equations with variables in X:

$$E = \{x = t_x(X) : x \in X\}.$$

For all $x \in X$, we have that $t_x(X)$ is an $ACP_{\tau,u}$-term with variables from the set X.

Definition (3.2.15)

Let t be a term over $ACP_{\tau,u}$ without abstracting operators. Suppose that in t a variable x occurs. We will call an occurrence of x in t guarded if t has a subterm of the form $a \cdot s$, in which a is an atomic action and s is a term over $ACP_{\tau,u}$, which contains this occurrence of x. Otherwise we will call the occurrence of x in t unguarded. We will call an $ACP_{\tau,u}$-term t without abstracting operators guarded if all occurrences of all variables in t are guarded. Let $E = \{x = t_x(X) : x \in X\}$ be a recursive specification without abstracting operators. We will call E a guarded recursive specification if we can rewrite it to a recursive specification E' with the aid of the axioms and/or the aid of the specification E itself in which all right-hand sides of the recursion equations of E' are guarded. We will call E' a completely guarded recursive specification.

Definition (3.2.16)

Let $E = \{x = t_x(X) : x \in X\}$ be a recursive specification. A solution for E is a vector $p = (p_x)_{x \in X}$, with $p_x \in P$ for all $x \in X$, such that for all $x \in X$ the following expressions are true statements: $p_x = t_x(p)$, in which $t_x(p)$ is shorthand for: substitute for each occurrence of an element $x \in X$ in $t_x(X)$ the process p_x. We say that two solutions are equal if the components of the vectors are equal: $(p_x)_{x \in X} = (q_x)_{x \in X}$ if and only if we have for each $x \in X$ that $p_x = q_x$.

Principle (3.2.17) *The Recursive Definition Principle*

Let E be a guarded recursive specification in the sense of definition (3.2.15). Then there is a solution for E. We will call this *RDP* but mostly it is called RDP^-.

Principle (3.2.18) *The Recursive Specification Principle*

Let E be a guarded recursive specification in the sense of definition (3.2.15). Then there is at most one solution for E.

Remark (3.2.19)

Now we will explain why we have excluded all abstracting operators in definition (3.2.15). First, we will show that we have to exclude the abstraction operator itself. Consider the following recursive specification:

$$E = \{\, x = i \cdot \tau_{\{j\}}(y),\ y = j \cdot \tau_{\{i\}}(x) \,\}.$$

Let there be an atomic action $a \in A \backslash \{i, j\}$, then it is easy to see that $(i{\cdot}a^n, j{\cdot}a^n)$ is a solution for E, for each $n \geq 1$. This means that we have to exclude the abstraction operator since *RSP* cannot be valid. The abstracting operators are of our next concern: with the aid of *ODP* it is very easy to make operators that behave like the abstraction operator. Consider the following linear functional specification for the set of names $N = \{n, m\}$.

$$\begin{aligned} E(N) = \{\, & n(i) = j, n(j) = i \,\} \\ \cup \{\, & n(a) = a : a \in A \setminus \{i, j\}\} \\ \cup \{\, & m(i) = \tau, m(j) = \tau \,\} \\ \cup \{\, & m(a) = a : a \in A \setminus \{i, j\}\} \\ \cup \{\, & n(i \cdot x) = j \cdot m(x), n(j \cdot x) = i \cdot m(x) \,\} \\ \cup \{\, & n(a \cdot x) = a \cdot n(x) : a \in A \setminus \{i, j\}\} \\ \cup \{\, & m(i \cdot x) = \tau \cdot n(x), m(j \cdot x) = \tau \cdot n(x) \,\} \\ \cup \{\, & m(a \cdot x) = a \cdot m(x) : a \in A \setminus \{i, j\}\}. \end{aligned}$$

According to *ODP*, there is a valuation $\varphi : N \longrightarrow F$, which solves the system $E(N)$. And by *OSP* there is at most one such a solution, so we can give these linear unary operators a name. Let us say, $\varphi(n) = \nu$ and $\varphi(m) = \mu$. Now consider the guarded recursive specification below:

$$E = \{\, x = i \cdot \nu(y),\ \ y = j \cdot \nu(x) \,\}.$$

Observe that we can prove the following for ν with induction on k:

$$\begin{aligned} \nu(a^k) &= a^k, \\ \nu(i^k) &= j^{\left[\frac{k+1}{2}\right]}, \\ \nu(j^k) &= i^{\left[\frac{k+1}{2}\right]}. \end{aligned}$$

Now it is easy to see that $(i^2 \cdot a^n, j^2 \cdot a^n)$ is a solution for this system E, for some $a \in A \setminus \{i, j\}$ and $n \in \mathbb{N}$. For:

$$\begin{aligned} i \cdot \nu(y) &= i \cdot \nu(j^2 \cdot a^n) \\ &= i^2 \cdot \mu(j \cdot a^n) \\ &= i^2 \cdot \tau \cdot \nu(a^n) \\ &= i^2 \cdot a^n \\ &= x. \end{aligned}$$

The calculation that $j \cdot \nu(x) = y$, is proved as above. This means that we found more than one solution for the guarded recursive specification above; hence, we have to exclude the abstracting operators, too, in the definition of a guarded recursive specification.

Observation (3.2.20)

In the above, we have just seen that if we want to use ACP_τ with renaming operators as they are defined in [**40**], we have to adjust the definition of a guarded recursive specification. For, we have to, at the least, exclude all the renaming operators ρ, such that $\rho(a) = \tau$, for some atomic action $a \in A$ in the definition of a guarded recursive specification.

Subsequently we will introduce a more restrictive version of the approximation induction principle that is called AIP^-. We will base ourselves on an idea that can be found in [**21**] but we will use a more simple version of AIP^- than the one proposed in [**21**].

Principle (3.2.21) *The Approximation Induction Principle*

Let x, y be $ACP_{\tau,u}$-terms. If we have for all $n \geq 1 : \pi_n(x) = \pi_n(y)$ and we have that x or y can be specified with the aid of a guarded recursive specification in the sense of definition (3.2.15), we have $x = y$.

Remark (3.2.22)

At present, we are able to explain why we demanded that x or y can be specified with the aid of a guarded recursive specification. It was pointed out in [**21**] that

$$ACP + RDP + AIP + CA + KFAR \vdash \tau = \tau + \tau \cdot \delta. \tag{2}$$

In other words the combination of these axioms is inconsistent. To solve this problem a more restrictive form of the approximation induction principle (AIP^-) is proposed there. We will use this idea in a simplified way: we require that x or y can be specified with the aid of a guarded recursive specification. The abbreviation CA stands for conditional axioms; they can be found in section 1.5. We will recall the particular conditional axiom $CA6$ that has been used to prove equation (2):

$$\forall J_1, J_2 \subseteq A \Longrightarrow \tau_{J_1} \circ \tau_{J_2} = \tau_{J_1 \cup J_2}.$$

With the aid of OSP it is very easy to prove this axiom. This will be done in corollary (3.4.25).

3.3. Termination

In this section we will prove the termination of a term rewriting system, associated with the axiom system $ACP_{\tau,u}$. We will exclude the axioms of commutativity of the alternative composition (A1) and the second and third τ-laws of Milner (T2–T3). First we will describe what exactly are closed $ACP_{\tau,u}$-terms.

Definition (3.3.1)

A closed term, or a closed $ACP_{\tau,u}$-term, is a term without variables of sort P and without variables of sort F.

Examples (3.3.2)

Suppose that we have just two atomic actions: $A = \{a, b\}$. We know that $\pi_1(a)$ is a closed term. We can rewrite this closed term to a term in which no elements of sort F occur. We see at once that $\pi_1(a)$ rewrites to a. Now let s be a function name. Consider the following linear functional specification.

$$\begin{aligned} E(s) = \{&s(a) = b, s(b) = a\} \\ &\cup \{s(a \cdot x) = b \cdot s(x), s(b \cdot x) = a \cdot s(x)\}. \end{aligned} \tag{1}$$

It is clear that $s(a^3)$ is a closed term. We can also rewrite this term with the aid of the linear functional specification $E(N)$. It is immediately clear that $s(a^3) = b^3$.

Remark (3.3.3)

We see that we can write the closed $ACP_{\tau,u}$-terms, considered above, without constants of sort F. This is what we want to prove. We will consider the term rewriting system associated with $ACP_{\tau,u}$ (see table 3.2) and we will also consider the equations of an arbitrary but fixed linear functional specification as rewriting rules from left to right. Thereinafter, we will study the termination of this whole system. The outline of this proof will be more or less the same as the proof of the termination of ACP_τ in appendix A of [**12**]. However, we will use the lexicographical variant of the recursive path ordering to prove the termination. The lexicographical variant of the recursive path ordering can be found in [**28**]. A general reference to the subject of term rewriting is [**20**]. Another reference that can be useful is [**30**]. The following definition is taken from [**12**], with some additions specific to the present situation.

Definition (3.3.4)

Let x and y be terms, let $\alpha \in A \cup C$ and let $f \in F$. Then we define the *weight* of a closed term as follows:

(i) $|\alpha| = 1$,

(ii) $|x \star y| = |x| + |y|$ for $\star = \cdot, \|, \mathbin{|\!\lfloor}, |$,

(iii) $|x + y| = \max\{|x|, |y|\}$,

(iv) $|\chi(f, x)| = |f(x)| = |x|$.

Definition (3.3.5) *The Partial Ordering of the Signature*

Because of problems with the reduction of the left-merge, the authors of [**12**] have introduced infinitely many operators: the so-called ranked operators. The rank of an operator $\|, \lfloor\!\lfloor, |$ is the weight of the subterm of which it is the leading operator. We will give the partial ordering of the operators. Let $m \geq 1$ and $n \geq 2$.

$$\|_n > \lfloor\!\lfloor_n, |_n \qquad \lfloor\!\lfloor_{n+1}, |_{n+1} > \|_n \qquad \chi, \|_n, \lfloor\!\lfloor_n, |_n > \cdot > + \qquad \pi_{m+1} > \pi_m.$$

Let N be a finite set of function names. The rank of a constant $n \in N$ is the weight of the subterm of which it is the leading operator. The set of these constants of rank k is denoted by N_k. The set of all these ranked constants of sort F is simply the union of all the N_k. We will denote this set by $N_r = \bigcup_{k=1}^{\infty} N_k$. We will give the partial ordering of these constants of sort F:

$$\forall\, n_k, m_l \in N_r : \quad k > l \Longrightarrow n_k > m_l.$$

For the remaining part of the signature we will define the partial ordering as follows. For all $n \in N_r$ and for all $a \in A$ we have: $n, |_2 > a > \tau > \delta$. See figure 3.2 for a graphical representation of this ordering.

Definition (3.3.6)

For each closed $ACP_{\tau,u}$-term t we obtain the ranked term t_r by assigning to all operators their rank.

Example (3.3.7)

Here we will use the constant s that we defined in equation (1). Let

$$t = \big(s(a) \lfloor\!\lfloor a \cdot b\big) \parallel \big(\tau \cdot s(a \cdot b) \mid s(a+b) \cdot a^2\big)$$

be a closed $ACP_{\tau,u}$-term. This term will be ranked as follows:

$$t_r = \big(s_1(a) \lfloor\!\lfloor_3\, a \cdot b\big) \parallel_{10} \big(\tau \cdot s_2(a \cdot b) \mid_7\ s_2(a+b) \cdot a^2\big). \tag{2}$$

Definition (3.3.8)

Let D be the set of ranked closed $ACP_{\tau,u}$-terms. Let D^* be D where some of the symbols may be marked with an asterisk $(*)$. As an example let us take the ranked closed $ACP_{\tau,u}$-term t_r considered above in equation (2). Then we have that a typical element $t_r^* \in D^*$ is the following expression:

$$t_r^* = \big(s_1^*(a) \lfloor\!\lfloor_3^*\, a \cdot b^*\big) \parallel_{10} \big(\tau^* \cdot s_2(a \cdot^* b) \mid_7\ s_2(a^* + b) \cdot a^2\big).$$

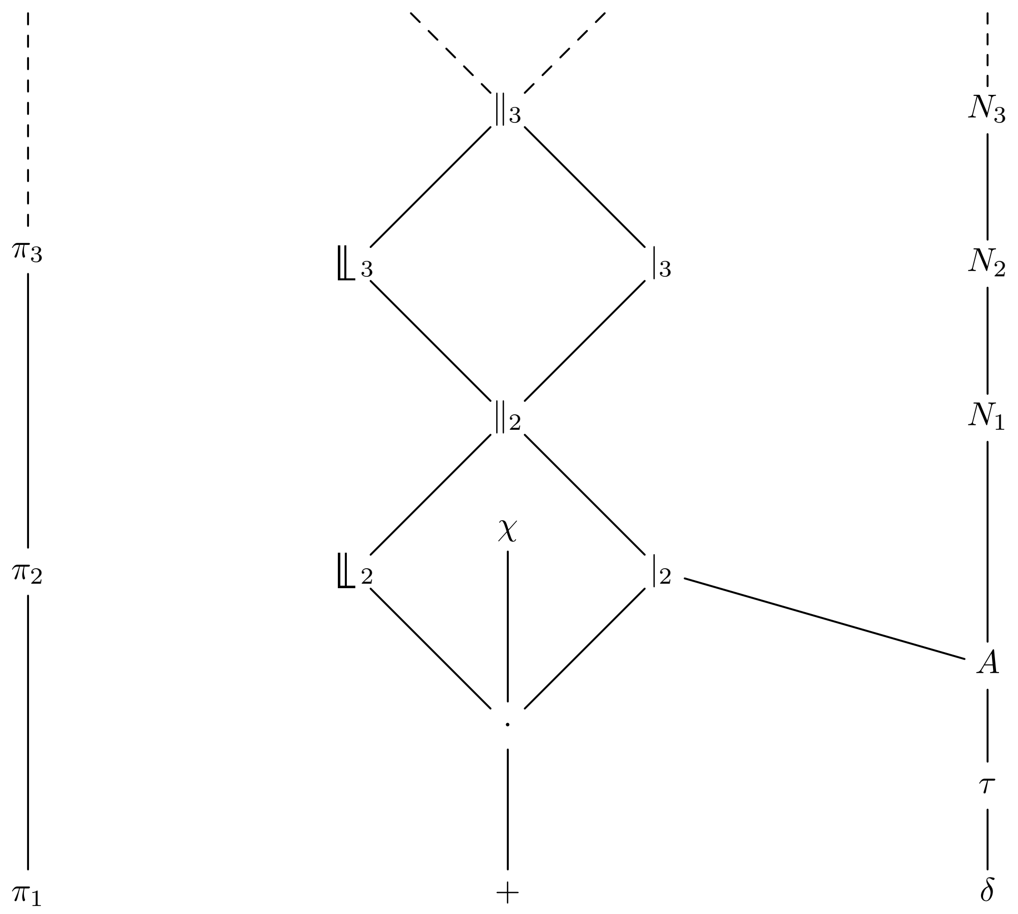

Figure 3.2. Visualization of the partial ordering.

Definition (3.3.9)

We will define a reduction relation "$\rightarrow$" on the set of marked ranked closed $ACP_{\tau,u}$-terms D^* as follows. For the sake of simplicity we will use, for the moment, prefix notation for the operators. Let H, G be function symbols occurring in the signature of $ACP_{\tau,u}$ (with this, we also mean constants such as atomic actions, δ, τ, or function names). Let $s, t, t_1, \ldots, t_k, s_1, \ldots, s_l$ be elements of D^*.

(i) $\quad H(t_1, \ldots, t_k) \rightarrow H^*(t_1, \ldots, t_k), \quad (k \geq 0)$

(ii) $\quad H^*(t_1, \ldots, t_k) \rightarrow G\big(H^*(t_1, \ldots, t_k), \ldots, H^*(t_1, \ldots, t_k)\big)$
$\qquad (H > G,\ k \geq 0)$

(iii) $\quad H^*(t_1, \ldots, t_k) \rightarrow t_i, \quad (k \geq 1,\ 1 \leq i \leq k)$

(iv) $\quad H^*\big(t_1, \ldots, G(s_1, \ldots, s_l), \ldots, t_k\big) \rightarrow H\big(t_1, \ldots, G^*(s_1, \ldots, s_l), \ldots, t_k\big)$
$\qquad (k \geq 1,\ l \geq 0)$

(v) $\quad s \rightarrow t \Longrightarrow H(\ldots, s, \ldots) \rightarrow H(\ldots, t, \ldots)$

(vi) $\quad$ If $t \equiv H^*\big(G(s_1, \ldots, s_l), t_2, \ldots, t_k\big)$, then $t \rightarrow H\big(G^*(s_1, \ldots, s_l), t, \ldots, t\big)$

$(k \geq 1,\ l \geq 0)$.

In which the ordering "$>$" on the signature of $ACP_{\tau,u}$ is already defined in definition (3.3.5). We will use the symbol "$\succ$" for the transitive closure of the above defined reduction relation.

RA2	$(x + y) + z \to x + (y + z)$	$x \cdot \tau \to x$	RT
RA3	$x + x \to x$		
RA4	$(x + y) \cdot z \to x \cdot z + y \cdot z$	$\tau \mathbin{\|\!\lfloor} x \to \tau \cdot x$	RTM1
RA5	$(x \cdot y) \cdot z \to x \cdot (y \cdot z)$	$(\tau \cdot x) \mathbin{\|\!\lfloor} y \to \tau \cdot (x \parallel y)$	RTM2
RA6	$x + \delta \to x$	$\tau \mid x \to \delta$	RTC1
RA7	$\delta \cdot x \to \delta$	$x \mid \tau \to \delta$	RTC2
		$(\tau \cdot x) \mid y \to x \mid y$	RTC3
RC	$a \mid b \to c_{a,b}$	$x \mid (\tau \cdot y) \to x \mid y$	RTC4
RCM1	$x \parallel y \to x \mathbin{\|\!\lfloor} y + (y \mathbin{\|\!\lfloor} x + x \mid y)$	$\chi(\tau_I, a) \to a, \quad \text{if } a \notin I$	RTI1
RCM2	$a \mathbin{\|\!\lfloor} x \to a \cdot x$	$\chi(\tau_I, a) \to \tau, \quad \text{if } a \in I$	RTI2
RCM3	$(a \cdot x) \mathbin{\|\!\lfloor} y \to a \cdot (x \parallel y)$	$\chi(\tau_I, x \cdot y) \to \chi(\tau_I, x) \cdot \chi(\tau_I, y)$	RTI3
RCM4	$(x + y) \mathbin{\|\!\lfloor} z \to x \mathbin{\|\!\lfloor} z + y \mathbin{\|\!\lfloor} z$		
RCM5	$(a \cdot x) \mid b \to (a \mid b) \cdot x$	$\chi(f \circ g, x) \to \chi\big(f, \chi(g, x)\big)$	RXC1
RCM6	$a \mid (b \cdot x) \to (a \mid b) \cdot x$	$\chi\big((f \circ g) \circ h, x\big) \to \chi\big(f \circ (g \circ h), x\big)$	RXC2
RCM7	$(a \cdot x) \mid (b \cdot y) \to (a \mid b) \cdot (x \parallel y)$	$\chi(f, \gamma) \to \gamma$	RX1
RCM8	$(x + y) \mid z \to x \mid z + y \mid z$	$\chi(f, \gamma \cdot x) \to \gamma \cdot \chi(f, x)$	RX2
RCM9	$x \mid (y + z) \to x \mid y + x \mid z$	$\chi(f, x + y) \to \chi(f, x) + \chi(f, y)$	RX3
RD1	$\chi(\partial_H, a) \to a, \quad \text{if } a \notin H$	$\chi(\pi_n, a) \to a$	RPR1
RD2	$\chi(\partial_H, a) \to \delta, \quad \text{if } a \in H$	$\chi(\pi_1, a \cdot x) \to a$	RPR2
RD3	$\chi(\partial_H, x \cdot y) \to \chi(\partial_H, x) \cdot \chi(\partial_H, y)$	$\chi(\pi_{n+1}, a \cdot x) \to a \cdot \chi(\pi_n, x)$	RPR3

Table 3.2. A term rewriting system associated with $ACP_{\tau,u}$.

Definition (3.3.10)

A partially ordered set $(S, >)$ consists of a set and a transitive and irreflexive binary relation $>$ defined on the elements of S. Notice that asymmetry of such a *strict* partial ordering follows from transitivity and irreflexivity. A partially ordered set $(S, >)$ is said to be *well-founded* if there are no infinite (strictly) descending sequences $s_1 > s_2 > s_3 > \cdots$ of elements of S.

Definition (3.3.11)

A term rewriting system over a set of terms has the termination property, if no infinite derivations are possible. A derivation is a sequence of rewrites.

It is known that the following theorem holds. See [**20**].

Theorem (3.3.12) *Dershowitz*

Let (Σ, R) be a term rewriting system with finitely many rewriting rules and let "$>$" be a well-founded ordering on Σ. If $s \succ t$ for each rewriting rule $s \to t \in R$, then the term rewriting system (Σ, R) has the termination property. Where the arrow in $s \to t$ is, of course, not the arrow that we defined in definition (3.3.9), but ordinary notation for a rewriting rule.

At this point we are about to discuss the table of rewriting rules, associated with the axiom system $ACP_{\tau,u}$. See table 3.2 at page 83. In this table, we use the same notational conventions as in table 3.1. We have a rewriting rule "RC" instead of making rewriting rules of the axioms C1–C3 in table 3.1. In fact, we describe in these axioms some properties of the predefined communication function. We give in RC a "listing" of all the function applications, so we can rewrite the communication-merge in case we have a term containing an expression $a \mid b$, with $a, b \in A_\delta (= A \cup \delta)$. We have no rewriting rules that correspond to the axioms T2–T3. We have done this because these axioms do not have a clear direction: they can be used in both directions in order to simplify a certain term. Consider the following reduction:

$$\begin{aligned} \tau \cdot (a + b) + a &= \tau \cdot (a + b) + a + b + a \\ &= \tau \cdot (a + b) + a + b \\ &= \tau \cdot (a + b). \end{aligned}$$

In this example, we use twice the second τ-law of Milner; in both directions. With the third τ-law of Milner, the same difficulty arises.

Definition (3.3.13)

In remark (3.3.3) we already announced that we will add to the term rewriting system associated with $ACP_{\tau,u}$, a finite set of rewrite rules. Thence, let N be a set of function names for a linear functional specification $E(N)$. Recall that $E(N)$ is of the following form.

$$E(N) = \left\{ r_{n,a} \mid n \in N, a \in A \right\} \cup \left\{ e_{n,a} \mid n \in N, a \in A \right\}.$$

We will make this system of equations into a term rewriting system from left to right by simply substituting for an equality sign $=$ an arrow $\to$ in the boundary conditions and the functional equations. We will call the set of all these rules: the term rewriting system associated with the linear functional specification $E(N)$.

Theorem (3.3.14)

Let $E(N)$ be a linear functional specification. The rewriting rules in table 3.2 on page 83 together with the term rewriting system associated with the linear functional specification $E(N)$ have the termination property. See definition (3.3.11).

Proof. According to theorem (3.3.12), it is sufficient to prove, for each closed instance $s \to t$ of the rewriting rules that $s \succ t$. Let us first take a closer look at the rewriting rule RA2. We will make use of (vi).

$$\begin{aligned}(x+y)+z &\succ (x+y)+^{*} z\\ &\succ (x+^{*} y)+\big((x+y)+^{*} z\big)\\ &\succ x+\big((x+^{*} y)+z\big)\\ &\succ x+(y+z).\end{aligned}$$

Now let us treat RA3.

$$\begin{aligned}x+x &\succ x+^{*} x\\ &\succ x.\end{aligned}$$

This means that $x+x \succ x$. Now $\cdot > +$ so we find for RA4:

$$\begin{aligned}(x+y)\cdot z &\succ (x+y)\cdot^{*} z\\ &\succ (x+y)\cdot^{*} z+(x+y)\cdot^{*} z\\ &\succ (x+^{*} y)\cdot z+(x+^{*} y)\cdot z\\ &\succ x\cdot z+y\cdot z.\end{aligned}$$

Indeed, $(x+y)\cdot z \succ x\cdot z+y\cdot z$. Now we will treat RA5. We will use (vi), too. In fact, this case is proved analogously to RA2.

$$\begin{aligned}(x\cdot y)\cdot z &\succ (x\cdot y)\cdot^{*} z\\ &\succ (x\cdot^{*} y)\cdot\big((x\cdot y)\cdot^{*} z\big)\\ &\succ x\cdot\big((x\cdot^{*} y)\cdot z\big)\\ &\succ x\cdot(y\cdot z).\end{aligned}$$

So we see that $(x\cdot y)\cdot z \succ x\cdot(y\cdot z)$. The rewrite rules RA6 and RA7 are proved exactly the same as RA3. So let us verify RC. We make use of the fact that the communication-merge of rank two is greater than all atomic actions. Observe that we also have $|_2 > \delta$. We are to show that $a \mid_2 b \succ c_{a,b}$, with $c_{a,b} \in A_\delta$.

$$\begin{aligned}a \mid_2 b &\succ a \mid_2^{*} b\\ &\succ c_{a,b}.\end{aligned}$$

Now let us take a closer look at the merge. First we will handle RCM1. Let $|x| + |y| = n$. Notice that we are to show:

$$x \parallel_n y \succ x \mathbin{\lfloor\!\lfloor}_n y + (y \mathbin{\lfloor\!\lfloor}_n x + x \mid_n y).$$

We will make use of the fact that $\parallel_n > +$ and that $\parallel_n > \mathbin{\lfloor\!\lfloor}_n, \mid_n$.

$$\begin{aligned}
x \parallel_n y &\succ x \parallel_n^* y \\
&\succ x \parallel_n^* y + x \parallel_n^* y \\
&\succ x \parallel_n^* y + (x \parallel_n^* y + x \parallel_n^* y) \\
&\succ (x \parallel_n^* y) \mathbin{\lfloor\!\lfloor}_n (x \parallel_n^* y) + \big((x \parallel_n^* y) \mathbin{\lfloor\!\lfloor}_n (x \parallel_n^* y) + (x \parallel_n^* y) \mid_n (x \parallel_n^* y)\big) \\
&\succ x \mathbin{\lfloor\!\lfloor}_n y + (y \mathbin{\lfloor\!\lfloor}_n x + x \mid_n y).
\end{aligned}$$

We will verify RCM2. Let $|x| = n$.

$$\begin{aligned}
a \mathbin{\lfloor\!\lfloor}_{n+1} x &\succ a \mathbin{\lfloor\!\lfloor}_{n+1}^* x \\
&\succ \big(a \mathbin{\lfloor\!\lfloor}_{n+1}^* x\big) \cdot \big(a \mathbin{\lfloor\!\lfloor}_{n+1}^* x\big) \\
&\succ a \cdot x.
\end{aligned}$$

Now we will handle the case RCM3. We will make use of the ranking of the operators. Let $|x| + |y| = n$. Then we easily find:

$$\begin{aligned}
(a \cdot x) \mathbin{\lfloor\!\lfloor}_{n+1} y &\succ (a \cdot x) \mathbin{\lfloor\!\lfloor}_{n+1}^* y \\
&\succ \big((a \cdot x) \mathbin{\lfloor\!\lfloor}_{n+1}^* y\big) \cdot \big((a \cdot x) \mathbin{\lfloor\!\lfloor}_{n+1}^* y\big) \\
&\succ (a \cdot x) \cdot \Big(\big((a \cdot x) \mathbin{\lfloor\!\lfloor}_{n+1}^* y\big) \parallel_n \big((a \cdot x) \mathbin{\lfloor\!\lfloor}_{n+1}^* y\big)\Big) \\
&\succ (a \cdot^* x) \cdot \big((a \cdot x) \parallel_n y\big) \\
&\succ a \cdot \big((a \cdot^* x) \parallel_n y\big) \\
&\succ a \cdot (x \parallel_n y).
\end{aligned}$$

We will consider RCM4. Let $|x| + |y| + |z| = p$, $|x| + |z| = q$ and $|y| + |z| = r$. We want to deduce that:

$$(x + y) \mathbin{\lfloor\!\lfloor}_p z \succ x \mathbin{\lfloor\!\lfloor}_q z + y \mathbin{\lfloor\!\lfloor}_r z.$$

Now contemplate the following calculation.

$$\begin{aligned}
(x + y) \mathbin{\lfloor\!\lfloor}_p z &\succ (x + y) \mathbin{\lfloor\!\lfloor}_p^* z \\
&\succ \big((x + y) \mathbin{\lfloor\!\lfloor}_p^* z\big) + \big((x + y) \mathbin{\lfloor\!\lfloor}_p^* z\big) \\
&\succ \big((x +^* y) \mathbin{\lfloor\!\lfloor}_p z\big) + \big((x +^* y) \mathbin{\lfloor\!\lfloor}_p z\big) \\
&\succ x \mathbin{\lfloor\!\lfloor}_p z + y \mathbin{\lfloor\!\lfloor}_p z \\
&\succ x \mathbin{\lfloor\!\lfloor}_p^* z + y \mathbin{\lfloor\!\lfloor}_p^* z \\
&\succ \big((x \mathbin{\lfloor\!\lfloor}_p^* z) \mathbin{\lfloor\!\lfloor}_q (x \mathbin{\lfloor\!\lfloor}_p^* z)\big) + \big((y \mathbin{\lfloor\!\lfloor}_p^* z) \mathbin{\lfloor\!\lfloor}_r (y \mathbin{\lfloor\!\lfloor}_p^* z)\big) \\
&\succ x \mathbin{\lfloor\!\lfloor}_q z + y \mathbin{\lfloor\!\lfloor}_r z.
\end{aligned}$$

Indeed, we find that $(x+y) \Vert\!\!\lfloor_p z \succ x \Vert\!\!\lfloor_q z + y \Vert\!\!\lfloor_r z$. Let us take a look at RCM5. Let $|x| = n$. Then we are to show: $(a \cdot x) \mid_{n+2} b \succ (a \mid_2 b) \cdot x$. Consider thereto the display below:

$$\begin{aligned}
(a \cdot x) \mid_{n+2} b &\succ (a \cdot x) \mid^*_{n+2} b \\
&\succ \big((a \cdot x) \mid^*_{n+2} b\big) \cdot \big((a \cdot x) \mid^*_{n+2} b\big) \\
&\succ \big((a \cdot^* x) \mid_{n+2} b\big) \cdot (a \cdot x) \\
&\succ (a \mid_{n+2} b) \cdot (a \cdot^* x) \\
&\succ (a \mid^*_{n+2} b) \cdot x \\
&\succ \big((a \mid^*_{n+2} b) \mid_2 (a \mid^*_{n+2} b)\big) \cdot x \\
&\succ (a \mid_2 b) \cdot x.
\end{aligned}$$

The deduction for the rewriting rule RCM6 is the same as the deduction above. So let us verify RCM7. Let $|x| + |y| = n$.

$$\begin{aligned}
(a \cdot x) \mid_{n+2} (b \cdot y) &\succ (a \cdot x) \mid^*_{n+2} (b \cdot y) \\
&\succ \big((a \cdot x) \mid^*_{n+2} (b \cdot y)\big) \cdot \big((a \cdot x) \mid^*_{n+2} (b \cdot y)\big) \\
&\succ \Big(\big((a \cdot x) \mid^*_{n+2} (b \cdot y)\big) \mid_2 \big((a \cdot x) \mid^*_{n+2} (b \cdot y)\big)\Big) \\
&\qquad \cdot \Big(\big((a \cdot x) \mid^*_{n+2} (b \cdot y)\big) \Vert_n \big((a \cdot x) \mid^*_{n+2} (b \cdot y)\big)\Big) \\
&\succ \big((a \cdot x) \mid_2 (b \cdot y)\big) \cdot \big((a \cdot x) \Vert_n (b \cdot y)\big) \\
&\succ \big((a \cdot^* x) \mid_2 (b \cdot^* y)\big) \cdot \big((a \cdot^* x) \Vert_n (b \cdot^* y)\big) \\
&\succ (a \mid_2 b) \cdot (x \Vert_n y).
\end{aligned}$$

We will treat RCM8. Let $p = |x| + |y| + |z|$, $q = |x| + |z|$ and $r = |y| + |z|$. Observe that we must show that: $(x+y) \mid_p z \succ x \mid_q z + y \mid_r z$. Consider the following:

$$\begin{aligned}
(x+y) \mid_p z &\succ (x+y) \mid^*_p z \\
&\succ \big((x+y) \mid^*_p z\big) + \big((x+y) \mid^*_p z\big) \\
&\succ \big((x +^* y) \mid_p z\big) + \big((x +^* y) \mid_p z\big) \\
&\succ x \mid_p z + y \mid_p z \\
&\succ x \mid^*_p z + y \mid^*_p z \\
&\succ \big((x \mid^*_p z) \mid_q (x \mid^*_p z)\big) + \big((y \mid^*_p z) \mid_r (y \mid^*_p z)\big) \\
&\succ x \mid_q z + y \mid_r z.
\end{aligned}$$

RCM9 is treated analogously. Let us calculate RD1 and RD2. This can be done in one calculation. We will use the fact that for all atomic actions $a \in A : a > \delta$.

$$\begin{aligned}
\chi(\partial_H, a) &\succ \chi^*(\partial_H, a) \\
&\succ a \\
&\succ a^* \\
&\succ \delta.
\end{aligned}$$

We see that $\chi(\partial_H, a) \succ a \succ \delta$, so with this, we handled both RD1 and RD2. We will treat RD3.

$$\begin{aligned}\chi(\partial_H, x \cdot y) &\succ \chi^*(\partial_H, x \cdot y)\\ &\succ \chi^*(\partial_H, x \cdot y) \cdot \chi^*(\partial_H, x \cdot y)\\ &\succ \chi(\partial_H, x \cdot^* y) \cdot \chi(\partial_H, x \cdot^* y)\\ &\succ \chi(\partial_H, x) \cdot \chi(\partial_H, y).\end{aligned}$$

RT is proved just like, e.g., RA7. RTM1 goes like RCM2. RTM2 is as RCM3. Now we will show RTC1. We will use that $\tau > \delta$. Let $|x| = n$.

$$\begin{aligned}\tau \mid_{n+1} x &\succ \tau \mid^*_{n+1} x\\ &\succ \tau\\ &\succ \tau^*\\ &\succ \delta.\end{aligned}$$

Of course RTC2 goes the same. We will show RTC3. Let $n = |x| + |y|$.

$$\begin{aligned}(\tau \cdot x) \mid_{n+1} y &\succ (\tau \cdot x) \mid^*_{n+1} y\\ &\succ (\tau \cdot^* x) \mid_{n+1} y\\ &\succ x \mid_{n+1} y\\ &\succ x \mid^*_{n+1} y\\ &\succ \big(x \mid^*_{n+1} y\big) \mid_n \big(x \mid^*_{n+1} y\big)\\ &\succ x \mid_n y.\end{aligned}$$

So we find $(\tau \cdot x) \mid_{n+1} y \succ x \mid_n y$. The proof of RTC4 is the same. RTI1–3 are proved in the same way as RD1–3. Observe that we use that for all atomic actions $a \in A : a > \tau$ in RTI1–2. Let us take RXC1. In this deduction we will make use of (vi) for the function symbol χ.

$$\begin{aligned}\chi(f \circ g, x) &\succ \chi^*(f \circ g, x)\\ &\succ \chi\big(f \circ^* g, \chi^*(f \circ g, x)\big)\\ &\succ \chi\big(f, \chi(f \circ^* g, x)\big)\\ &\succ \chi\big(f, \chi(g, x)\big).\end{aligned}$$

To deduce the desired inequality for RXC2, we make use of (vi) for the function symbol $\circ$.

$$\begin{aligned}\chi\big((f \circ g) \circ h, x\big) &\succ \chi^*\big((f \circ g) \circ h, x\big)\\ &\succ \chi\big((f \circ g) \circ^* h, x\big)\\ &\succ \chi\big((f \circ^* g) \circ \big((f \circ g) \circ^* h\big), x\big)\\ &\succ \chi\big(f \circ \big((f \circ^* g) \circ h\big), x\big)\\ &\succ \chi\big(f \circ (g \circ h), x\big).\end{aligned}$$

The case RX1 is trivial. So let us treat RX2.

$$\begin{aligned}\chi(f, \gamma \cdot x) &\succ \chi^*(f, \gamma \cdot x)\\ &\succ \chi^*(f, \gamma \cdot x) \cdot \chi^*(f, \gamma \cdot x)\\ &\succ (\gamma \cdot x) \cdot \chi(f, \gamma \cdot^* x)\\ &\succ (\gamma \cdot^* x) \cdot \chi(f, x)\\ &\succ \gamma \cdot \chi(f, x).\end{aligned}$$

For RX3 we will make use of the fact that $\chi > +$.

$$\begin{aligned}\chi(f, x + y) &\succ \chi^*(f, x + y)\\ &\succ \chi^*(f, x + y) + \chi^*(f, x + y)\\ &\succ \chi(f, x +^* y) + \chi(f, x +^* y)\\ &\succ \chi(f, x) + \chi(f, y).\end{aligned}$$

RPR1 goes like, e.g., RD1. Let us verify RPR2.

$$\begin{aligned}\chi(\pi_1, a \cdot x) &\succ \chi^*(\pi_1, a \cdot x)\\ &\succ a \cdot x\\ &\succ a \cdot^* x\\ &\succ a.\end{aligned}$$

For the verification of RPR3 we will use that $\pi_{n+1} > \pi_n$.

$$\begin{aligned}\chi(\pi_{n+1}, a \cdot x) &\succ \chi^*(\pi_{n+1}, a \cdot x)\\ &\succ \chi^*(\pi_{n+1}, a \cdot x) \cdot \chi^*(\pi_{n+1}, a \cdot x)\\ &\succ (a \cdot x) \cdot \chi(\pi^*_{n+1}, a \cdot x)\\ &\succ (a \cdot^* x) \cdot \chi(\pi_n, a \cdot x)\\ &\succ a \cdot \chi^*(\pi_n, a \cdot x)\\ &\succ a \cdot \chi(\pi_n, x)\end{aligned}$$

Let us now take a boundary condition. Recall that we have introduced the rank of a function name, thus for the boundary condition we are to show $\chi(n_1, a) \succ b$. We will make use of the fact that $n_1 > b$.

$$\begin{aligned}\chi(n_1, a) &\succ \chi^*(n_1, a)\\ &\succ n_1\\ &\succ n_1^*\\ &\succ b.\end{aligned}$$

Indeed, we see that $\chi(n_1, a) \succ b$. Let us take a functional equation. Let $|x| = p$. We will show that $\chi(n_{p+1}, a \cdot x) \succ b \cdot \chi(m_p, x)$.

$$\begin{aligned}
\chi(n_{p+1}, a \cdot x) &\succ \chi^*(n_{p+1}, a \cdot x) \\
&\succ \chi^*(n_{p+1}, a \cdot x) \cdot \chi^*(n_{p+1}, a \cdot x) \\
&\succ n_{p+1} \cdot \chi(n_{p+1}, a \cdot^* x) \\
&\succ n_{p+1}^* \cdot \chi(n_{p+1}, x) \\
&\succ b \cdot \chi^*(n_{p+1}, x) \\
&\succ b \cdot \chi(n_{p+1}^*, x) \\
&\succ b \cdot \chi(m_p, x).
\end{aligned}$$

And finally, we find $\chi(n_{p+1}, a \cdot x) \succ b \cdot \chi(m_p, x)$. This ends the proof of 3.3.14.

Remarks (3.3.15)

The partial ordering on the signature in [**12**] differs in two ways from the partial ordering that is given in (3.3.5), or equivalently in figure 3.2. Firstly, we give the following ordering on atomic actions: for all $a \in A$, we defined $a > \tau > \delta$. In [**12**], there is no ordering on the atomic actions, nor on δ or τ, whatsoever. The author did not succeed in proving that $\partial_H(a) \succ \delta$, $\tau \mid x \succ \delta$, $x \mid \tau \succ \delta$ or $\tau_I(a) \succ \tau$, without some kind of ordering on the atomic actions. Secondly, in [**12**] there is no rewriting rule for $a \mid b$, with $a, b \in A \setminus \{\delta\}$. This is necessary in order to be able to prove an elimination result. This is also stated in [**12**], but without a rewriting rule for $a \mid b$, this result in [**12**] is not entirely correct.

We have seen that this way of proving termination is highly usable in process algebra. For, the method presented in [**12**], is generalized effortlessly to the present situation. Therefore, it is worthwhile investigating this method separately. However, we will not do that in this thesis. For another way of proving termination in process algebra we refer to [**1**].

Here we will discuss an elimination theorem which states that we can eliminate in a closed $ACP_{\tau,u}$-term all the operators that are not the alternative composition or the sequential composition. In other words, we can rewrite each closed $ACP_{\tau,u}$-term to a $BPA_{\delta,\tau}$-term. The acronym *BPA* stands for basic process algebra. This system consists of the first five laws of *PA*, which has been studied in [**14**]. The abbreviation *PA* stands for process algebra. The subscripts δ and τ, mean that the axioms concerning those special constants are added to the theory *BPA*. A good general reference to *BPA*, $BPA_{\delta,\tau}$ and *PA* is [**8**].

We will use the same notational conventions as before in table 3.1. For the axioms of $BPA_{\delta,\tau}$, see table 3.3 on page 91.

A1 $x + y = y + x$		$x \cdot \tau = x$ T1
A2 $x + (y + z) = (x + y) + z$		$\tau \cdot x + x = \tau \cdot x$ T2
A3 $x + x = x$		$a \cdot (\tau \cdot x + y) = a \cdot (\tau \cdot x + y) + a \cdot x$ T3
A4 $(x + y) \cdot z = x \cdot z + y \cdot z$		
A5 $(x \cdot y) \cdot z = x \cdot (y \cdot z)$		
A6 $x + \delta = x$		
A7 $\delta \cdot x = \delta$		

Table 3.3. $BPA_{\delta,\tau}$.

Lemma (3.3.16)

Let t be a closed $BPA_{\delta,\tau}$-term. Let RA3–7 and RT from table 3.2 on page 83 be the term rewriting system associated with $BPA_{\delta,\tau}$, then we can rewrite the term t in one of the following forms:

$$u = \begin{cases} a, & \text{with } a \in A; \\ \delta; & \\ \tau; & \\ a \cdot v, & \text{with } a \in A_\tau \text{ and } v \text{ a closed } BPA_{\delta,\tau}\text{-term in normal form;} \\ v + w, & \text{with } v, w \text{ closed } BPA_{\delta,\tau}\text{-terms in normal form.} \end{cases}$$

Or, equivalently, there are closed $BPA_{\delta,\tau}$-terms $x_1, \ldots, x_n, t_1, \ldots, t_p$ such that

$$u = \sum_{i=1}^{n} a_i \cdot x_i + \sum_{k=1}^{p} \tau \cdot t_k,$$

for certain atomic actions $a_1, \ldots, a_n \in A$ and $n, p \geq 0$.

Proof. This is a well-known fact. See, e.g., [**8**]. Note that we rewrite modulo A1 and A2. We did not do that in theorem (3.3.14); we excluded the axiom A1 among others in there in order to be able to use the method of the recursive path ordering.

Theorem (3.3.17) *The Elimination Theorem*

Let t be a closed $ACP_{\tau,u}$-term. Then there is a closed $BPA_{\delta,\tau}$-term s, such that

$$ACP_{\tau,u} \vdash t = s.$$

Proof. Let N be the set of function names that occur in the closed $ACP_{\tau,u}$-term t. Rewrite this term t with the aid of the term rewriting system in table 3.2

on page 83 and the term rewriting system associated with $E(N)^*$ to a normal form. With the aid of theorem (3.3.14), we know that this is not an infinite process, i.e., there is a finite row:

$$t = t_0 \to t_1 \to \cdots \to t_n = s, \tag{3}$$

and we cannot perform any rewriting rule on s. We immediately see that in s we have not a merge operator, for otherwise we could apply RCM1. We will distinguish five cases. If in s occurs a function name, a left-merge, a communication-merge, an encapsulation operator or an abstraction operator, we will take a minimal subterm (in the sense of a minimal number of symbols), in which precisely one of these operators occurs.

1 We have a minimal subterm of the form $n(u)$, with $n \in N$ and u a closed $BPA_{\delta,\tau}$-term. Then we know that u has one of the forms displayed in (3.3.16). If $u = a$, then we can use a boundary condition: $\chi(n, a) \to b$, but this is in contradiction with the assumption that s is in normal form. If $u = \delta$ or τ, we can apply RX1. If $u = a \cdot v$, we can use a functional equation: $\chi(n, a \cdot v) \to b \cdot \chi(m, v)$. And if $u = v + w$, we can apply RX3.

2 We have a minimal subterm of the form $u_1 \mathbin{⫗} u_2$, with u_1, u_2 closed $BPA_{\delta,\tau}$-terms. If $u_1 = a \in A_\delta$, then we can apply RCM2. If $u_1 = \tau$, we can use RTM1. If $u_1 = a \cdot v$, we can use RCM3. If $u_1 = v + w$, then we can use RCM4.

3 We have a minimal subterm of the form $u_1 \mid u_2$ with u_1, u_2 closed $BPA_{\delta,\tau}$-terms. If $u_1 = a \in A$ or δ, then we may use RC, RCM6 or RCM9. If $u_1 = \tau$, we can apply RTC1. If $u_1 = a \cdot v$, we can use RCM5 or RCM7 or RCM9 or RTC2. If $u_1 = v + w$, we can use RCM8.

4 We have a minimal subterm of the form $\partial_H(u)$, with u a closed $BPA_{\delta,\tau}$-term. If $u = a \in A$ or δ, we can apply RD1 or RD2. If $u = \tau$, we can apply RDT. If $u = a \cdot v$, we can use RD4 and, if $u = v + w$, we can use RD3.

5 We have a minimal subterm of the form $\tau_I(u)$, with u a closed $BPA_{\delta,\tau}$-term. This case is treated exactly the same as the former case.

So in s a left-merge, a communication-merge, an encapsulation operator or an abstraction operator cannot occur. Thus, s is a closed $BPA_{\delta,\tau}$-term. If we replace the arrows by equality signs, in equation (3), we obtain a proof in $ACP_{\tau,u}$ of $t = s$. This will end the proof of 3.3.17.

Corollary (3.3.18)

Let t be a closed $ACP_{\tau,u}$-term. Then we can rewrite this term t into a closed term u which has the following form:

$$u = \sum_{i=1}^{n} a_i \cdot x_i + \sum_{k=1}^{p} \tau \cdot t_k,$$

* see definition (3.3.13)

for certain closed $BPA_{\delta,\tau}$-terms $x_1, \ldots, x_n, t_1, \ldots, t_p$ with $n, p \geq 0$ and certain $a_1, \ldots, a_n \in A$.

3.4. Theorems

In several applications in which auxiliary linear unary operators have been used in the past, there was a need to deduce some basic properties of these operators in order to be able to prove the desired results. We can think of an application wherein the approximation induction principle is used to establish a result. See definition (1.2.6) for the formulation of *AIP*. Therefore, we often need to know whether or not this particular auxiliary operator commutes with the projection operators π_n. In this section we will prove that if f can be defined with the aid of a linear functional specification and, if f is a concrete operator, then for all processes x that contain no τs, we have

$$\pi_n \circ f(x) = f \circ \pi_n(x).$$

See theorem (3.4.8) for details. Moreover, this yields that if we work in the axiom system ACP_u (ACP_u is $ACP_{\tau,u}$ without abstraction; see table 3.5 on page 115, section 3.5), all the linear unary operators that can be defined with the aid of a linear functional specification, commute with the projection operators. For, in that case all the operators and processes are both concrete.

Another thing that we might want to know is which processes are fixed points of such an operator. We will prove two results about this. To formulate these theorems we will use the notion of stable atomic actions (with respect to this operator). We call an atomic action stable with respect to $f \in F$ if we have for all $x \in P$ that $f(a \cdot x) = a \cdot f(x)$, that is, if f meets such an atom, it will pass it and nothing will change: neither the atomic action, nor the operator itself.

It is intuitively clear that, for instance, $\rho \circ \rho = \rho$ for a renaming operator that only renames $a \in A$ into b. This can be proved within the framework of ACP_τ, for closed terms. But we actually feel that this must be valid for open terms. With the aid of *OSP*, we can prove this for open terms. In fact, we will prove some theorems on idempotent linear unary operators that can be defined with the aid of a linear functional specification.

It is not only the case that we might want to know some basic facts about auxiliary unary operators that we defined in the middle of a verification, but it can be the case that we want to know something concerning operators that we already know. For instance, it is immediately clear that $\tau_I \circ \tau_{\{i\}} = \tau_I$, for all $i \in I$. Again it is already possible to prove this for closed terms within ACP_τ. With the aid of *OSP* it is very trivial to show that this is true in general.

We see that the operator $\tau_{\{i\}}$ is absorbed by τ_I. We will define the notions of a left and right-absorber. We will prove some theorems concerning this matter.

In [**5**] we can find conditional axioms. They are of great interest for algebraic verification techniques. We have listed these conditional axioms in section 1.5. We will treat a theorem from which two of these axioms follow immediately; namely CA5 and CA6. Just another thing that can be of interest, is the question if a linear unary operator commutes with the encapsulation operator, or with the abstraction operator. Or when do we have that ∂_H and τ_I commute? The latter question is also known as a conditional axiom CA7. In fact, we will prove this conditional axiom CA7 as a corollary of theorem (3.4.24), in which we give some necessary conditions in a way that renaming operators commute. We extend this result to arbitrary linear unary operators that can be defined with the aid of a linear functional specification. In these theorems we will use the notion of stable atomic actions, too.

Finally, we will treat, for "historic" reasons, a theorem that gives the conditions such that

$$f \circ g(x \parallel y) = f(x) \parallel g(y).$$

This theorem can be found in [**4**]. We included it, since it can be seen as one of the first general theorems on auxiliary linear unary operators. This theorem might look a bit strange at first sight, but when we take $f = g = \partial_H$, we find a very useful equation: $\partial_H(x \parallel y) = \partial_H(x) \parallel \partial_H(y)$, since ∂_H is idempotent. (Of course, we do have certain restrictions on x and y.)

Hereinafter, we will define the notion of the alphabet of a process, which can be found in [**5**]. In section 1.5 we already defined this notion. However, we have to adapt this definition to the present situation.

Definition (3.4.1)

Let x be a closed $ACP_{\tau,u}$-term and let $a \in A$. The alphabet of a process x is the set of atomic actions that x can perform. We define inductively what the alphabet $\alpha(x)$ of this x is. If in x occurs a constant of sort F, we will rewrite this term x with corollary (3.3.18) to a term x' without constants of sort F and we define $\alpha(x) := \alpha(x')$. Now let x be a closed $ACP_{\tau,u}$-term without elements of sort F.

(i)	$\alpha(\delta) = \alpha(\tau) = \emptyset$
(ii)	$\alpha(a) = \{a\}$
(iii)	$\alpha(\delta \cdot x) = \emptyset$
(iv)	$\alpha(\tau \cdot x) = \alpha(x)$
(v)	$\alpha(a \cdot x) = \{a\} \cup \alpha(x)$
(vi)	$\alpha(x + y) = \alpha(x) \cup \alpha(y)$

Now we have defined the alphabet for closed terms, we will define it for terms t, with the property that $\pi_n(t)$ is a closed $ACP_{\tau,u}$-term.

(vii) $\qquad \alpha(t) = \bigcup_{n=1}^{\infty} \alpha\big(\pi_n(t)\big)$

Definition (3.4.2)

Let $f \in F$ be a unary operator. An atomic action $a \in A$ is called *stable* (with respect to f), if we have for all $x \in P$:

$$f(a \cdot x) = a \cdot f(x).$$

Otherwise, it is called *unstable*. The set of all stable atomic actions with respect to f is denoted:

$$S(f) = \big\{\, a \in A \,\big|\, \forall x \in P : f(a \cdot x) = a \cdot f(x) \,\big\}.$$

We will use the notation $U(f) = A \setminus S(f)$ for the set of unstable atomic actions with respect to f.

Examples (3.4.3)

Let $I \subseteq A$. Then the set of unstable atomic actions of the abstraction operator is $U(\tau_I) = I$. Let $H \subseteq A$. Then the set of unstable atomic actions of the encapsulation operator is $U(\partial_H) = H$. Let $n \geq 1$. Then the set of stable atomic actions of the projection operator is $S(\pi_n) = \emptyset$.

Remark (3.4.4)

Let $f \in F$ be a linear unary operator. Let $a \in A$ be a stable atom with respect to f. Then we find that $f(a) = a$. For consider the following:

$$\begin{aligned} f(a) &= f(a \cdot \tau) \\ &= a \cdot f(\tau) \\ &= a \cdot \tau \\ &= a. \end{aligned}$$

Definition (3.4.5)

A process $x \in P$ is called concrete, if it has the following form:

$$x = \sum_{i=1}^{n} a_i \cdot x_i + \sum_{j=1}^{m} b_j,$$

for $a_1, \ldots, a_n, b_1, \ldots, b_m \in A_\delta$ and $x_1, \ldots, x_n \in P$ are concrete.

Theorem (3.4.6)

Let x be a concrete process, then for all $n \geq 1$, we have that $\pi_n(x)$ is a closed $ACP_{\tau,u}$-term.

Proof. We will prove 3.4.6 with induction on n. We know that x is of the following form:

$$x = \sum_{i=1}^{n} a_i \cdot x_i + \sum_{j=1}^{m} b_j,$$

and $x_1, \ldots, x_n$ are also concrete (by definition). Now let $n = 1$, then

$$\pi_1(x) = \sum_{i=1}^{n} a_i + \sum_{j=1}^{m} b_j,$$

and this is a closed term. Suppose that 3.4.6 is proved for n, then we prove it for $n + 1$. We know that for all $i = 1, \ldots, n : \pi_n(x_i) = t_i$ are closed terms, so we find for x:

$$\begin{aligned}\pi_{n+1}(x) &= \sum_{i=1}^{n} a_i \cdot \pi_n(x_i) + \sum_{j=1}^{m} b_j \\ &= \sum_{i=1}^{n} a_i \cdot t_i + \sum_{j=1}^{m} b_j,\end{aligned}$$

which is a closed $ACP_{\tau,u}$-term.

Remark (3.4.7)

Notice that we can define the alphabet of a concrete process with the aid of the former theorem, for we defined the alphabet inductively on closed $ACP_{\tau,u}$-terms and the projections of concrete processes are closed terms.

Theorem (3.4.8)

Let $f \in F$ be a concrete linear unary operator that can be defined with a linear functional specification (see definition (3.2.12) for the definition of a concrete operator). Let x be a concrete process. Then we have for all $n \geq 1$:

$$\pi_n \circ f(x) = f \circ \pi_n(x).$$

Proof. We will prove this theorem with induction on n. Recall that x has the following form:

$$x = \sum_{i=1}^{n} a_i \cdot x_i + \sum_{j=1}^{m} b_j,$$

for certain concrete $x_1, \ldots, x_n$. Let $n = 1$, then,

$$\begin{aligned}
\pi_1 \circ f(x) &= \sum_{i=1}^{n} f(a_i) + \sum_{j=1}^{m} f(b_j) \\
&= \sum_{i=1}^{n} f\big(\pi_1(a_i \cdot x_i)\big) + \sum_{j=1}^{m} f\big(\pi_1(b_j)\big) \\
&= f \circ \pi_1(x).
\end{aligned}$$

Suppose that 3.4.8 is valid for n, then we prove it for $n+1$. Observe that all $g \in D(f)$ are concrete, if f itself is concrete.

$$\begin{aligned}
\pi_{n+1} \circ f(x) &= \sum_{i=1}^{n} f(a_i) \cdot \pi_n \circ g_i(x_i) + \sum_{j=1}^{m} \pi_{n+1} \circ f(b_j) \\
&= \sum_{i=1}^{n} f(a_i) \cdot g_i \circ \pi_n(x_i) + \sum_{j=1}^{m} f(b_j) \\
&= \sum_{i=1}^{n} f\big(a_i \cdot \pi_n(x_i)\big) + \sum_{j=1}^{m} f\big(\pi_{n+1}(b_j)\big) \\
&= f \circ \pi_{n+1}(x).
\end{aligned}$$

Since $g_i \in D(f)$. This finishes the proof of theorem 3.4.8.

Lemma (3.4.9)

Let $f \in F$ be a linear unary operator, not necessarily definable by a linear functional specification, and let x be a closed $ACP_{\tau,u}$-term. Then we have the following:

$$\alpha(x) \subseteq S(f) \Longrightarrow f(x) = x.$$

Proof. We will prove 3.4.9 with induction on the number n of symbols of x. So let $n = 1$, then we have three possibilities for x: $x = a$, $x = \delta$, or $x = \tau$. Because of X1, the latter two cases are proved and, due to the fact that $\alpha(a) \subseteq S(f)$ and remark (3.4.4), we know that $f(a) = a$. Now let $n > 1$ and suppose that 3.4.9 holds for all closed terms with their number of symbols $< n$. Because of (3.3.18), we know that x has the following form:

$$x = \sum_{i=1}^{n} a_i \cdot x_i + \sum_{k=1}^{p} \tau \cdot t_k,$$

for closed $ACP_{\tau,u}$-terms $x_1, \ldots, x_n, t_1, \ldots, t_p$, $n, p \geq 0$ and atomic actions $a_1, \ldots, a_n \in A$. Now we see that $a_i \in \alpha(x) \subseteq S(f)$, hence, $f(a_i \cdot x_i) = a_i \cdot f(x_i)$, for $1 \leq i \leq n$. Consider the following:

$$\begin{aligned} f(x) &= \sum_{i=1}^{n} f(a_i \cdot x_i) + \sum_{k=1}^{p} f(\tau \cdot t_k) \\ &= \sum_{i=1}^{n} a_i \cdot f(x_i) + \sum_{k=1}^{p} \tau \cdot f(t_k) \\ &= \sum_{i=1}^{n} a_i \cdot x_i + \sum_{k=1}^{p} \tau \cdot t_k \\ &= x. \end{aligned}$$

Observe that we can use the induction hypothesis, since

$$\alpha(x_i), \alpha(t_k) \subseteq \alpha(x) \subseteq S(f).$$

This ends the proof of 3.4.9.

Theorem (3.4.10)

Let $f \in F$ be a concrete linear unary operator that can be defined with a linear functional specification and let x be a concrete process, and suppose that $\alpha(x) \subseteq S(f)$, then x is a fixed point of f.

Proof. According to *AIP* it suffices to prove for all $n \geq 1$:

$$\pi_n \circ f(x) = \pi_n(x).$$

Let $n \geq 1$ be fixed. As both x and f are concrete, we know that

$$\pi_n \circ f(x) = f \circ \pi_n(x) = f\big(\pi_n(x)\big).$$

With the aid of (3.4.6) and (3.4.9) we know that $f\big(\pi_n(x)\big) = \pi_n(x)$, since

$$\alpha\big(\pi_n(x)\big) \subseteq \bigcup_{i=1}^{\infty} \alpha\big(\pi_i(x)\big) = \alpha(x) \subseteq S(f).$$

Thus, we see that x is a fixed point of f.

Definition (3.4.11)

Let $f \in F$ be a linear unary operator. If we have $f^2 = f$, we will call f idempotent.

Theorem (3.4.12)

Let $f \in F$ be a renaming operator. Suppose that the following condition holds.

$$\forall\, \alpha \in f\big(U(f)\big) : f(\alpha) = \alpha,$$

then f is idempotent, that is, $f^2 = f$.

Proof. Let n be a function name and consider the following linear functional specification.

$$\begin{aligned} E(n) = \{\, & n(a) = f(a) : a \in A \,\} \\ \cup \{\, & n(a \cdot x) = n(a) \cdot n(x) : a \in A \,\}. \end{aligned}$$

We immediately see that f is a solution for this system. Now we will show that f^2 is also a solution for it. Let $a \in S(f)$, then we see that $f^2(a) = a = f(a)$. Let $a \in U(f)$, then we know that $f(a)$ is a fixed point of f. So we find again $f^2(a) = f(a)$. Since f is a renaming, we find $f^2(a \cdot x) = f^2(a) \cdot f^2(x)$. We see that f^2 is a solution for the linear functional specification $E(n)$. But according to *OSP*, we know that there is at most one solution, so we find $f^2 = f$. This ends the proof of 3.4.12.

Theorem (3.4.13)

Let $f_1 \in F$ be a linear unary operator that can be defined with the aid of a linear functional specification. Let $D(f_1) = \{f_1, \ldots, f_k\}$ be the set of derived operators. Let

$$\sigma : A \times \{1, \ldots, k\} \longrightarrow \{1, \ldots, k\}$$

be defined as follows. If we have for all $x \in P : f_i(a \cdot x) = f_i(a) \cdot f_j(x)$, then we define $\sigma(a, i) = j$. Suppose that the following conditions hold,

(*i*) $\quad \forall\, \alpha \in f_i\big(U(f_i)\big) : f_i(\alpha) = \alpha$

(*ii*) $\quad \sigma(a, i) = \sigma\big(f_i(a), i\big)$

then f_i is idempotent, for all $1 \leq i \leq k$.

Proof. Let $N = \{n_1, \ldots, n_k\}$ be a set of function names. Consider the following linear functional specification.

$$\begin{aligned} E(N) = \{\, & n_i(a) = f_i(a) : a \in A,\, 1 \leq i \leq k \,\} \\ \cup \{\, & n_i(a \cdot x) = n_i(a) \cdot n_{\sigma(a,i)}(x) : a \in A,\, 1 \leq i \leq k \,\}. \end{aligned}$$

It will be clear that $f_1, \ldots, f_k$ is a solution for this system of equations. We will show that $f_1^2, \ldots, f_k^2$ is also a solution for this system. Choose an $i \in \{1, \ldots, k\}$. As in theorem (3.4.12), we see at once that $f_i^2(a) = f_i(a)$. We will handle the functional equations. Here, we will use the second condition.

$$\begin{aligned} f_i^2(a \cdot x) &= f_i\big(f_i(a) \cdot f_{\sigma(a,i)}(x)\big) \\ &= f_i^2(a) \cdot f_{\sigma(f_i(a),i)} \circ f_{\sigma(a,i)}(x) \\ &= f_i^2(a) \cdot f_{\sigma(a,i)}^2(x). \end{aligned}$$

With the aid of *OSP*, we see that $f_i^2 = f_i$, for all $1 \le i \le k$. This will end the proof of 3.4.13.

Definition (3.4.14)

Let $f, g \in F$. If $f \circ g = g$, we will call g a left-absorber for f; if $g \circ f = g$, we say that g is a right-absorber for f. If g is a left-absorber and a right-absorber for f, we just say that g is an absorber for f.

Theorem (3.4.15) *(Left-absorption)*

Let $f, g \in F$ be renaming operators. If we have

$$S(g) \cup g\big(U(g)\big) \subseteq S(f) \cup \{\delta, \tau\},$$

then g is a left-absorber for f.

Proof. We are to show that $f \circ g = g$. Let n be a function name. Consider the following linear functional specification.

$$E(n) = \big\{\, n(a) = g(a) : a \in A \,\big\} \cup \big\{\, n(a \cdot x) = n(a) \cdot n(x) : a \in A \,\big\}.$$

We immediately see that g is a solution for $E(n)$. It is very easy to deduce that $f \circ g$ is also a solution for it. Hence, with the aid of *OSP*, we find that $f \circ g = g$, so g is a left-absorber for f. This will end the proof of 3.4.15.

Theorem (3.4.16) *(Left-absorption)*

Let $f, g_1 \in F$. Suppose that f is a renaming operator and suppose that g_1 can be defined with the aid of a linear functional specification. Let $D(g_1) = \{g_1, \ldots, g_l\}$ be the set of derived operators of g_1. Suppose that the following condition holds:

$$\bigcap_{j=1}^{l} \Big(S(g_j) \cup g_j\big(U(g_j)\big) \Big) \subseteq S(f) \cup \{\delta, \tau\}.$$

Then all g_j are left-absorbers for f.

Proof. Let $N = \{n_1, \ldots, n_l\}$ be a set of function names. Define a map

$$\rho : A \times \{1, \ldots, l\} \longrightarrow \{1 \ldots, l\}$$

as follows:

$$\rho(a, j) = k \iff \forall\, x \in P : g_j(a \cdot x) = g_j(a) \cdot g_k(x).$$

Consider the following linear functional specification.

$$\begin{aligned} E(N) = {} & \big\{\, n_j(a) = g_j(a) : a \in A,\ 1 \le j \le l \,\big\} \\ & \cup \big\{\, n_j(a \cdot x) = n_j(a) \cdot n_{\rho(a,j)}(x) : a \in A,\ 1 \le j \le l \,\big\}. \end{aligned}$$

It will be clear that $g_1, \ldots, g_l$ is a solution for this system of equations. We will show that

$$f \circ g_1, \ldots, f \circ g_l$$

is also a solution. It is very easy to see that $f \circ g_j(a) = g_j(a)$. Hence, all $f \circ g_j$ satisfy the boundary conditions of $E(N)$. We will show that the functional equations are satisfied, as well.

$$\begin{aligned} f \circ g_j(a \cdot x) &= f\big(g_j(a) \cdot g_{\rho(a,j)}(x)\big) \\ &= f \circ g_j(a) \cdot f \circ g_{\rho(a,j)}(x). \end{aligned}$$

We find thus, according to *OSP*, that $f \circ g_j = g_j$, and all g_j are left-absorbers for f. This ends the proof.

Observe that in these two absorption theorems, we could have replaced the conditions respectively by $f \circ g(a) = g(a)$ and $f \circ g_j(a) = g_j(a)$. We did *not* do that for orthogonality reasons: with the conditions as they are, we can formulate the following generalization.

Theorem (3.4.17) *(Left-absorption)*

Let $f_1, g_1 \in F$ be definable with the aid of linear functional specifications. Let their sets of derived operators be as follows.

$$D(f_1) = \{f_1, \ldots, f_k\},\ D(g_1) = \{g_1, \ldots, g_l\}.$$

Suppose that the following condition holds.

$$\bigcap_{j=1}^{l} \Big(S(g_j) \cup g_j\big(U(g_j)\big)\Big) \subseteq \bigcap_{i=1}^{k} S(f_i) \cup \{\delta, \tau\}.$$

Then all g_j are left-absorbers for all f_i.

Proof. We will use the notations of (3.4.16). We must show that $f_i \circ g_j$ is a solution for $E(N)$. First, we will handle the boundary conditions. Fix $1 \leq i \leq k$ and $1 \leq j \leq l$. First, let $a \in S(g_j)$, then we see that $a \in S(f_i)$, so $f_i \circ g_j(a) = a = g_j(a)$. Now suppose that $a \in U(g_j)$. If $g_j(a) = \gamma$, with $\gamma \in \{\delta, \tau\}$, then we have

$$\begin{aligned} f_i \circ g_j(a) &= f_i(\gamma) \\ &= \gamma \\ &= g_j(a). \end{aligned}$$

If $g_j(a) \in S(f_i)$, then we find immediately that $f_i \circ g_j(a) = g_j(a)$. Hence, the boundary conditions are treated. Now we will handle the functional equations. We will only treat the case that $a \in U(g_j)$ and $g_j(a) \neq \gamma$. So $g_j(a) \in S(f_i)$. We see that

$$\begin{aligned} f_i \circ g_j(a \cdot x) &= f_i\big(g_j(a) \cdot g_{\rho(a,j)}(x)\big) \\ &= f_i \circ g_j(a) \cdot f_i \circ g_{\rho(a,j)}(x). \end{aligned}$$

Hence, we obtain with the aid of *OSP* that $f_i \circ g_j = g_j$ for all i and j. This is what we wanted to prove.

Theorem (3.4.18) *(Right-absorption)*

Let $f, g \in F$ be renaming operators. Suppose that

$$\forall\, a \in A : f \circ g(a) = f(a),$$

then f is a right-absorber for g.

Proof. Let n be a function name. Consider the following linear functional specification.

$$E(n) = \big\{\, n(a) = f(a) : a \in A \,\big\} \cup \big\{\, n(a \cdot x) = n(a) \cdot n(x) : a \in A \,\big\}.$$

We see that f is a solution for $E(n)$. It is trivial to deduce that the same holds for $f \circ g$, so with *OSP*, we find that $f \circ g = f$. This is what we wanted to prove.

Theorem (3.4.19) *(Right-absorption)*

Let $f_1, g \in F$. Suppose that f_1 is definable with the aid of a linear functional specification, and let g be a renaming operator. Let $D(f_1) = \{f_1, \ldots, f_k\}$ be the set of derived operators of f_1. Define $\sigma : A \times \{1, \ldots, k\} \longrightarrow \{1, \ldots, k\}$ as follows

$$\sigma(a, i) = j \iff \forall\, x \in P : f_i(a \cdot x) = f_i(a) \cdot f_j(x).$$

Suppose that the following conditions are valid.

(i) $\quad f_i \circ g(a) = f_i(a),\ 1 \le i \le k,\ a \in A$

(ii) $\quad \sigma\big(g(a), i\big) = \sigma(a, i),\ 1 \le i \le k,\ a \in A$

Then all f_i are right-absorbers for g.

Proof. Let $N = \{n_1, \ldots, n_k\}$ be a set of function names. Consider the linear functional specification hereinafter.

$$\begin{aligned} E(N) = \big\{\, n_i(a) = f_i(a) : a \in A, 1 \le i \le k \,\big\} \\ \cup \big\{\, n_i(a \cdot x) = n_i(a) \cdot n_{\sigma(a,i)}(x) : a \in A, 1 \le i \le k \,\big\}. \end{aligned}$$

We see that $f_1, \ldots, f_k$ is a solution for this system of equations. We will show that this is also valid for $f_i \circ g$ with $1 \le i \le k$. Because of the first condition we will only have to treat the functional equations.

$$\begin{aligned} f_i \circ g(a \cdot x) &= f_i\big(g(a) \cdot g(x)\big) \\ &= f_i \circ g(a) \cdot f_{\sigma(g(a),i)} \circ g(x) \\ &= f_i \circ g(a) \cdot f_{\sigma(a,i)} \circ g(x). \end{aligned}$$

Observe that we used the second condition. We find thus, with the aid of *OSP* that $f_i \circ g = f_i$. This is precisely what we wanted to prove.

Theorem 3.4.19 will not generalize any further. For, suppose that $g \in F$ is also definable with the aid of a linear functional specification and suppose that $|D(g)| > 1$. Then there is an atomic action $a \in A$ and a linear unary operator $h \in D(g)$, such that for all $x \in P : g(a \cdot x) = g(a) \cdot h(x)$. But then we find with the second condition that $f_i \circ g(a \cdot x) = f_i \circ g(a) \cdot f_{\sigma(a,i)} \circ h(x)$, but this functional equation does not correspond with any of the functional equations in the linear functional specification $E(N)$ which defines $f_1, \ldots, f_k$. So we find that $f_i \circ g$ is not a solution for it. It turns out that the assumption that $|D(g)| > 1$ cannot hold. We find thus that g must be a renaming operator.

Theorem (3.4.20)

Let $f, f_1, \ldots, f_k$ be renaming operators. Suppose that the following holds:

$$\forall\, a \in A : f(a) = f_1 \circ f_2 \circ \cdots \circ f_k(a)$$

then $f = f_1 \circ f_2 \circ \cdots \circ f_k$.

Proof. Let n be a function name. Consider the following linear functional specification.

$$E(n) = \left\{\, n(a) = f(a) : a \in A \,\right\} \cup \left\{\, n(a \cdot x) = n(a) \cdot n(x) : a \in A \,\right\}.$$

It will be clear that f is a solution for this system. But we also see that

$$f_1 \circ f_2 \circ \cdots \circ f_k$$

is a solution for it; so in accordance with *OSP* we may conclude that they are equal. This will end the proof of 3.4.20.

Corollary (3.4.21)

Let $H_1, H_2 \subseteq A$ and let $H = H_1 \cup H_2$. Then we have $\partial_H = \partial_{H_1} \circ \partial_{H_2}$.

Proof. It is trivial to verify that the conditions of theorem (3.4.20) are satisfied. With this, we conclude the proof of 3.4.21.

Corollary (3.4.22)

Let $I_1, I_2 \subseteq A$ and let $I = I_1 \cup I_2$. Then we have $\tau_I = \tau_{I_1} \circ \tau_{I_2}$.

Proof. Trivial.

Remarks (3.4.23)

Both corollaries (3.4.21) and (3.4.22) are known as conditional axioms; see section 1.5. We see that it is very trivial to prove these statements, with this theory. In the setting of ACP_τ it is only possible to prove these axioms for closed terms.

Theorem (3.4.24)

Let $f, g \in F$ be renaming operators. Suppose that the following holds:

(i) $\quad S(f) \cup S(g) = A,$

(ii) $\quad f\big(U(f)\big) \subseteq S(g) \cup \{\delta, \tau\},$

(iii) $\quad g\big(U(g)\big) \subseteq S(f) \cup \{\delta, \tau\},$

then f and g commute, i.e., $f \circ g = g \circ f$.

Proof. Let n be a function name. Consider the linear functional specification $E(n)$ below:

$$\begin{aligned} E(n) = &\left\{\, n(a) = a : a \in S(f) \cap S(g) \,\right\} \\ &\cup \left\{\, n(a) = f(a) : a \in S(g) \setminus S(f) \,\right\} \\ &\cup \left\{\, n(a) = g(a) : a \in S(f) \setminus S(g) \,\right\} \\ &\cup \left\{\, n(a \cdot x) = n(a) \cdot n(x) : a \in A \,\right\}. \end{aligned}$$

First, we will show that $f \circ g$ is a solution for the linear functional specification above. If a is in $S(f) \cap S(g)$, then we see that $f \circ g(a) = a$. Let $a \in S(g) \setminus S(f)$; then we see with (ii) that $f \circ g(a) = f(a)$. Let $a \in S(f) \setminus S(g)$; then we obtain with the aid of (iii) that $f \circ g(a) = g(a)$. Finally, we take $a \in A$ and $x \in P$; then we easily find that $f \circ g(a \cdot x) = f \circ g(a) \cdot f \circ g(x)$. This means that $f \circ g$ is a solution for the system $E(n)$. We can also show that $g \circ f$ is a solution for the linear functional specification above, so with the aid of *OSP*, we find that $f \circ g = g \circ f$. This ends the proof of 3.4.24.

Corollary (3.4.25)

Let $I, H \subseteq A$. Suppose that $I \cap H = \emptyset$. Then the encapsulation operator ∂_H and the abstraction operator τ_I commute.

Proof. We will verify the conditions of theorem (3.4.24). We know that $S(\partial_H) = A \setminus H$ and $S(\tau_I) = A \setminus I$, so because of the fact that $I \cap H = \emptyset$, we immediately see that (i) holds. We see that $\partial_H(H) = \{\delta\} \subseteq S(\tau_I) \cup \{\delta, \tau\}$, so (ii) is valid. The same applies to (iii), thus, we may use theorem (3.4.24) and we find that $\tau_I \circ \delta_H = \delta_H \circ \tau_I$. This ends the proof.

Remark (3.4.26)

Corollary (3.4.25) is also known as a conditional axiom; see section 1.5. In the setting of ACP_τ, it is possible to prove this for closed ACP_τ-terms, but in the framework of $ACP_{\tau,u}$, it is possible to prove this axiom for all processes.

Theorem (3.4.27)

Let $f \in F$ be a renaming operator. Let $k_1 \in F$ be a linear unary operator that can be defined with the aid of a linear functional specification. Let the set of derived operators of k_1 be $D(k_1) = \{k_1, \ldots, k_l\}$. Suppose that the following holds:

(i)	$S(f) \cup \bigcap_{i=1}^{l} S(k_i) = A$
(ii)	$f\big(U(f)\big) \subseteq \bigcap_{i=1}^{l} S(k_i) \cup \{\delta, \tau\}$
(iii)	$\bigcup_{i=1}^{l} k_i\big(U(k_i)\big) \subseteq S(f) \cup \{\delta, \tau\}$

then f and k_i commute, i.e., for all i in $\{1, \ldots, l\}$, we have: $f \circ k_i = k_i \circ f$.

Proof. Define a map $\sigma : A \times \{1, \ldots, l\} \longrightarrow \{1, \ldots, l\}$ as follows. Let $a \in A$ and $1 \leq i \leq l$ be chosen. We know that there is an operator $k_j \in D(k_1)$, such that $k_i(a \cdot x) = b \cdot k_j(x)$, for a certain $a \in A \cup C$. Now we will define $\sigma(a, i) = j$. Let $M = \{m_1, \ldots, m_l\}$ be a set of function names. Consider the following linear functional specification. Let $C = \bigcap_{i=1}^{l} S(k_i)$.

$$\begin{aligned}
E(M) = \big\{\, & m_i(a) = a : a \in S(f) \cap C,\ 1 \leq i \leq l \,\big\} \\
& \cup \big\{\, m_i(a) = k_i(a) : a \in S(f) \setminus C,\ 1 \leq i \leq l \,\big\} \\
& \cup \big\{\, m_i(a) = f(a) : a \in C \setminus S(f),\ 1 \leq i \leq l \,\big\} \\
& \cup \big\{\, m_i(a \cdot x) = a \cdot m_i(x) : a \in S(f) \cap C,\ 1 \leq i \leq l \,\big\} \\
& \cup \big\{\, m_i(a \cdot x) = k_i(a) \cdot m_{\sigma(a,i)}(x) : a \in S(f) \setminus C,\ 1 \leq i \leq l \,\big\} \\
& \cup \big\{\, m_i(a \cdot x) = f(a) \cdot m_i(x) : a \in C \setminus S(f),\ 1 \leq i \leq l \,\big\}.
\end{aligned}$$

Let $x \in P$ and $i \in \{1, \ldots, l\}$ be fixed. Let $a \in S(f) \cap C$, then we see that $k_i \circ f(a) = a$ and we see that $f \circ k_i(a) = a$, too. Moreover, we see that

$$k_i \circ f(a \cdot x) = a \cdot k_i \circ f(x)$$

and

$$f \circ k_i(a \cdot x) = a \cdot f \circ k_i(x).$$

Now let $a \in S(f) \setminus C$. Then we also see that $k_i \circ f(a) = k_i(a)$, and we see that $f \circ k_i(a) = k_i(a)$, because of *(iii)*. Moreover, we see the following:

$$\begin{aligned}
f \circ k_i(a \cdot x) &= f\big(k_i(a) \cdot k_{\sigma(a,i)}(x)\big) \\
&= k_i(a) \cdot f \circ k_{\sigma(a,i)}(x)
\end{aligned}$$

and

$$\begin{aligned}
k_i \circ f(a \cdot x) &= k_i\big(a \cdot f(x)\big) \\
&= k_i(a) \cdot k_{\sigma(a,i)} \circ f(x).
\end{aligned}$$

Now let $a \in C \setminus S(f)$. Then we see that $k_i \circ f(a) = f(a)$ and we see that $f \circ k_i(a) = f(a)$. It is also easy to see that

$$\begin{aligned} k_i \circ f(a \cdot x) &= k_i\big(f(a) \cdot f(x)\big) \\ &= f(a) \cdot k_i \circ f(x) \end{aligned}$$

and

$$\begin{aligned} f \circ k_i(a \cdot x) &= f\big(a \cdot k_i(x)\big) \\ &= f(a) \cdot f \circ k_i(x), \end{aligned}$$

since f is a renaming operator. Thus, we find that $k_1 \circ f, \ldots, k_l \circ f$ is a solution for the linear functional specification above. But we also see that $f \circ k_1, \ldots, f \circ k_l$ is a solution for this system. So with the aid of *OSP*, we may conclude that $k_i \circ f = f \circ k_i$. Herewith we end the proof of 3.4.27.

Theorem (3.4.28)

Let $f_1, g_1 \in F$ be linear unary operators that can be defined with the aid of linear functional specifications. Let the sets of derived operators be given as follows:

$$\begin{aligned} D(f_1) &= \{f_1, \ldots, f_n\}, \\ D(g_1) &= \{g_1, \ldots, g_m\}. \end{aligned}$$

Suppose that the following holds:

(i) $\bigcap_{i=1}^{n} S(f_i) \cup \bigcap_{j=1}^{m} S(g_j) = A$

(ii) $\bigcup_{i=1}^{n} f_i\big(U(f_i)\big) \subseteq \bigcap_{j=1}^{m} S(g_j) \cup \{\delta, \tau\}$

(iii) $\bigcup_{j=1}^{m} g_i\big(U(g_i)\big) \subseteq \bigcap_{i=1}^{n} S(f_i) \cup \{\delta, \tau\}$

then all elements of the derived operator set $D(f_1)$ of f_1, commute with all elements of the derived operator set $D(g_1)$ of g_1.

Proof. Define two maps

$$\begin{aligned} \sigma &: A \times \{1, \ldots, n\} \longrightarrow \{1, \ldots, n\} \\ \rho &: A \times \{1, \ldots, m\} \longrightarrow \{1, \ldots, m\} \end{aligned}$$

as follows. Let $a \in A$ and $1 \leq i \leq n$ be chosen. There is an $f_j \in D(f_1)$ such that $f_i(a \cdot x) = b \cdot f_j(x)$, for a certain $b \in A \cup C$. We define $\sigma(a, i) = j$. Now fix $c \in A$ and $1 \leq j \leq m$. We know that there is a $g_k \in D(g_1)$ such that $g_j(c \cdot x) = d \cdot g_k(x)$ for a certain $d \in A \cup C$. We define $\rho(c, j) = k$. Let $M = \{m_{i,j} : 1 \leq i \leq n, 1 \leq j \leq m\}$ be a set of function names. We will use the following abbreviations:

$$S_1 = \bigcap_{i=1}^{n} S(f_i) \text{ and } S_2 = \bigcap_{j=1}^{m} S(g_j).$$

Consider the linear functional specification hereinafter.

$$\begin{aligned}
E(M) = \big\{\, & m_{i,j}(a) = a : a \in S_1 \cap S_2,\ 1 \le i \le n, 1 \le j \le m \,\big\} \\
\cup \big\{\, & m_{i,j}(a) = g_j(a) : a \in S_1 \setminus S_2,\ 1 \le i \le n, 1 \le j \le m \,\big\} \\
\cup \big\{\, & m_{i,j}(a) = f_i(a) : a \in S_2 \setminus S_1,\ 1 \le i \le n, 1 \le j \le m \,\big\} \\
\cup \big\{\, & m_{i,j}(a \cdot x) = a \cdot m_{i,j}(x) : a \in S_1 \cap S_2,\ 1 \le i \le n, 1 \le j \le m \,\big\} \\
\cup \big\{\, & m_{i,j}(a \cdot x) = g_j(a) \cdot m_{i,\rho(a,j)}(x) \\
& : a \in S_1 \setminus S_2,\ 1 \le i \le n, 1 \le j \le m \,\big\} \\
\cup \big\{\, & m_{i,j}(a \cdot x) = f_i(a) \cdot m_{\sigma(a,i),j}(x) \\
& : a \in S_2 \setminus S_1,\ 1 \le i \le n, 1 \le j \le m \,\big\}.
\end{aligned}$$

Let $x \in P$, $1 \le i \le n$ and $1 \le j \le m$ be fixed. Choose $a \in S_1 \cap S_2$. We immediately see that $f_i \circ g_j(a) = a = g_j \circ f_i(a)$. It is also easy to see the following.

$$f_i \circ g_j(a \cdot x) = a \cdot f_i \circ g_j(x)$$

and

$$g_j \circ f_i(a \cdot x) = a \cdot g_j \circ f_i(x).$$

Now let $a \in S_1 \setminus S_2$. Then it is easy to see that $f_i \circ g_j(a) = g_j(a) = g_j \circ f_i(a)$. It is also immediately clear that

$$\begin{aligned}
f_i \circ g_j(a \cdot x) &= f_i\big(g_j(a) \cdot g_{\rho(a,j)}(x)\big) \\
&= g_j(a) \cdot f_i \circ g_{\rho(a,j)}(x)
\end{aligned}$$

and

$$\begin{aligned}
g_j \circ f_i(a \cdot x) &= g_j\big(a \cdot f_i(x)\big) \\
&= g_j(a) \cdot g_{\rho(a,j)} \circ f_i(x).
\end{aligned}$$

Observe that we used here (*iii*). Finally let $a \in S_2 \setminus S_1$. We see that $f_i \circ g_j(a) = f_i(a) = g_j \circ f_i(a)$. Moreover, using (*ii*), we find that

$$\begin{aligned}
f_i \circ g_j(a \cdot x) &= f_i\big(a \cdot g_j(x)\big) \\
&= f_i(a) \cdot f_{\sigma(a,i)} \circ g_j(x)
\end{aligned}$$

and

$$\begin{aligned}
g_j \circ f_i(a \cdot x) &= g_j\big(f_i(a) \cdot f_{\sigma(a,i)}(x)\big) \\
&= f_i(a) \cdot g_j \circ f_{\sigma(a,i)}(x).
\end{aligned}$$

Thus, we find that $\{f_i \circ g_j : 1 \le i \le n, 1 \le j \le m\}$ is a solution for $E(M)$. But we also find that $\{g_j \circ f_i : 1 \le i \le n, 1 \le j \le m\}$ is a solution for $E(M)$. So with the use of *OSP*, we find that $f_i \circ g_j = g_j \circ f_i$, for all $1 \le i \le n$ and $1 \le j \le m$. This ends the proof of 3.4.28.

Theorem (3.4.29)

Suppose that there is no communication, that is, $a \mid b = \delta$ for all atomic actions $a, b \in A$. Let $f_1, g_1 \in F$ be definable with the aid of linear functional specifications. Let their sets of derived operators be $D(f_1) = \{f_1, \ldots, f_n\}$ and $D(g_1) = \{g_1, \ldots, g_m\}$. Let x, y be closed $ACP_{\tau,u}$-terms. Suppose that the following conditions hold.

(i) $\quad \alpha(x) \cup \bigcup_{u=1}^{n} \alpha\big(f_u(x)\big) \subseteq \bigcap_{v=1}^{m} S(g_v)$

(ii) $\quad \alpha(y) \cup \bigcup_{v=1}^{m} \alpha\big(g_v(y)\big) \subseteq \bigcap_{u=1}^{n} S(f_u)$

Then we have for all $1 \leq u \leq n$ and $1 \leq v \leq m$:

$$f_u \circ g_v(x \parallel y) = f_u(x) \parallel g_v(y), \tag{1}$$

$$f_u \circ g_v(x \mathbin{\lfloor\!\lfloor} y) = f_u(x) \mathbin{\lfloor\!\lfloor} g_v(y). \tag{2}$$

Proof. First we will show that equation (1) is correct. With that result we will prove equation (2). Observe that this is not the "usual" order. This is caused by the fact that if we know that equation (2) holds, we do not know that

$$f_u \circ g_v(y \mathbin{\lfloor\!\lfloor} x) = g_v(y) \mathbin{\lfloor\!\lfloor} f_u(x).$$

We will prove (1) with induction on the sum n of the number of symbols of x and of y. First we will consider the basis of our induction: $n = 2$. We will have four possibilities:

$$x = a, y = b \quad x = a, y = \tau \quad x = \tau, y = b \quad x = y = \tau,$$

with $a, b \in A_\delta$. We will deduce only the first one. Let $1 \leq u \leq n$ and $1 \leq v \leq m$ be fixed. Consider the following.

$$\begin{aligned} f_u \circ g_v(a \parallel b) &= f_u \circ g_v(a \cdot b) + f_u \circ g_v(b \cdot a) \\ &= f_u(a) \cdot g_v(b) + g_v(b) \cdot f_u(a) \\ &= f_u(a) \parallel g_v(b). \end{aligned}$$

Now let $n \geq 2$, and suppose that (1) is correct for n. We will prove it for $n+1$. Let x, y be chosen. Recall that they can be written as $BPA_{\delta,\tau}$-terms:

$$\begin{aligned} x &= \sum_i a_i \cdot x_i + \sum_k \tau \cdot t_k, \\ y &= \sum_j b_j \cdot y_j + \sum_l \tau \cdot s_l. \end{aligned}$$

Let u_i be such that $f_u(a_i \cdot z) = f_u(a_i) \cdot f_{u_i}(z)$ and let v_j be such that $g_v(b_j \cdot z) = g_v(b_j) \cdot g_{v_j}(z)$. Consider the calculation below.

$$\begin{aligned}
f_u \circ g_v(x \parallel y) &= \sum_i f_u \circ g_v\big(a_i \cdot (x_i \parallel y)\big) + \sum_k \tau \cdot f_u \circ g_v(t_k \parallel y) \\
&\quad + \sum_j f_u \circ g_v\big(b_j \cdot (x \parallel y_j)\big) + \sum_l \tau \cdot f_u \circ g_v(x \parallel s_l) \\
&= \sum_i f_u\big(a_i \cdot g_v(x_i \parallel y)\big) + \sum_k \tau \cdot f_u \circ g_v(t_k \parallel y) \\
&\quad + \sum_j f_u\big(g_v(b_j) \cdot g_{v_j}(x \parallel y_j)\big) + \sum_l \tau \cdot f_u \circ g_v(x \parallel s_l) \\
&= \sum_i f(a_i) \cdot f_{u_i} \circ g_v(x_i \parallel y) + \sum_k \tau \cdot f_u \circ g_v(t_k \parallel y) \\
&\quad + \sum_j g_v(b_j) \cdot f_u \circ g_{v_j}(x \parallel y_j) + \sum_l \tau \cdot f_u \circ g_v(x \parallel s_l)
\end{aligned}$$

It will be clear that we may use the induction hypothesis four times.

$$\begin{aligned}
&= \sum_i f(a_i) \cdot \big(f_{u_i}(x_i) \parallel g_v(y)\big) + \sum_k \tau \cdot \big(f_u(t_k) \parallel g_v(y)\big) \\
&\quad + \sum_j g_v(b_j) \cdot \big(g_{v_j}(y_j) \parallel f_u(x)\big) + \sum_l \tau \cdot \big(g_v(s_l) \parallel f_u(x)\big) \\
&= f_u(x) \mathbin{\|\!_} g_v(y) + g_v(y) \mathbin{\|\!_} f_u(x) \\
&= f_u(x) \parallel g_v(y).
\end{aligned}$$

This will end the proof of equation (1). To deduce equation (2) we find immediately that

$$f_u \circ g_v(x \mathbin{\|\!\_} y) = \sum_i f_u(a_i) \cdot f_{u_i} \circ g_v(x_i \parallel y) + \sum_k \tau f_u \circ g_v(t_k \parallel y).$$

With the aid of equation (1), we see that this yields

$$\begin{aligned}
&= \sum_i f_u(a_i) \cdot \big(f_{u_i}(x_i) \parallel g_v(y)\big) + \sum_k \tau \cdot \big(f_u(t_k) \parallel g_v(y)\big) \\
&= f_u(x) \mathbin{\|\!_} g_v(y).
\end{aligned}$$

This will end the proof of 3.4.29.

Remark (3.4.30)

We treated theorem (3.4.29), as it can be seen as one of the first general theorems concerning linear unary operators. It can be found in [**4**]. It is stated in terms of the state operator; see section 1.3 and it uses the notion of the alphabet of an object to give the necessary conditions. This theorem, however, as it is stated in [**4**], is wrong. The definition of the alphabet of an object is "wrong". Even if we adjust this definition, the theorem still remains wrong. To exemplify this we will give hereinafter the definitions needed to formulate this theorem. Subsequently, we will state it and comment upon it. We will not have abstraction here, since it is not considered in [**4**]. The following definition is taken from [**4**].

We will give the definition of the alphabet $\alpha(m)$ of an object $m \in M$, as the set of all actions that can be changed, so

$$\alpha(m) = \{a \in A \mid \exists s \in S : a(m,s) \neq a\}.$$

The following is also copied from [**4**].

Theorem (3.4.31)
If there is no communication and $\alpha(x) \cap \alpha(m_1) = \alpha(y) \cap \alpha(m_2) = \emptyset$, then

$$\lambda_{s_1}^{m_1} \circ \lambda_{s_2}^{m_2}(x \parallel y) = \lambda_{s_1}^{m_1}(x) \parallel \lambda_{s_2}^{m_2}(y).$$

First of all we will show that this theorem is not correct. Let

$$M = \{m, m'\}, \quad S = \{s\}, \quad A = \{a, b, c, d\}.$$

We will give *act* and *eff*. We have $c(m,s) = d$ and $b(m',s) = c$. Further, nothing changes: $x(m,s) = x(m',s) = x$ for $x \in \{a, d\}$. Observe that for all $x \in A$:

$$s(m,x) = s(m',x) = s.$$

The conditions of 3.4.31 are satisfied for $x = a$ and $y = b$:

$$\alpha(a) \cap \alpha(m') = \{a\} \cap \{b\} = \emptyset,$$
$$\alpha(b) \cap \alpha(m) = \{b\} \cap \{c\} = \emptyset.$$

It is very easy to see that $\lambda_s^m \circ \lambda_s^{m'}(a \parallel b) = a \parallel d$, but we also see that

$$\lambda_s^m(a) \parallel \lambda_s^{m'}(b) = a \parallel c.$$

We will explain what is "wrong" with $\alpha(m)$, for $m \in M$. If we have the situation that $a(m,s) = a$, we can still have $s(m,a) \neq s$. So this action has changed the operator. For the alphabet of an object $m \in M$ we take the following set (this is the definition as it appears in [**8**], 6.4.10.4):

$$\alpha(m) = \{a \in A \mid \exists s \in S : a(m,s) \neq a\} \cup \{a \in A \mid \exists t \in S : t(m,a) \neq t\}.$$

With this definition the example presented above, is still a counterexample for 3.4.31. Observe that we have the following:

$$\alpha(m) = \bigcup_{s \in S} A \setminus S(\lambda_s^m). \tag{3}$$

Now if we adjust the conditions in 3.4.31, as is done below, we have a correct formulation of this theorem.

Theorem (3.4.32)

Let x, y be closed *ACP*-terms. Suppose that there is no communication. If we have for m_1, m_2 in M and s_1, s_2 in S the following

$$\Big[\alpha(x) \cup \bigcup_{s \in S} \alpha\big(\lambda_s^{m_1}(x)\big)\Big] \cap \alpha(m_2) = \emptyset,$$

$$\Big[\alpha(y) \cup \bigcup_{s \in S} \alpha\big(\lambda_s^{m_2}(y)\big)\Big] \cap \alpha(m_1) = \emptyset,$$

then we have for $\star = \|, \mathbin{⌊\!⌊}$:

$$\lambda_{s_1}^{m_1} \circ \lambda_{s_2}^{m_2}(x \star y) = \lambda_{s_1}^{m_1}(x) \star \lambda_{s_2}^{m_2}(y).$$

Observe that the conditions here are more or less the same as in (3.4.29). In fact, the conditions in here are stronger, since in general

$$\big\{\, t \in S : \lambda_t^m \in D(\lambda_s^m) \,\big\} \subsetneq S.$$

With the aid of equation (3), we transform easily from the "empty intersection conditions" to the form in (3.4.29).

Proof. This is left to the reader.

3.5. A Model

In this section we will not construct a model for $ACP_{\tau,u}$, but we will "forget" about all the axioms concerning τs. Thus, we will confine ourselves to an axiom system that is called ACP_u. See table 3.5 on page 115. We will construct for ACP_u the standard model of process algebra: the projective limit model. We will do this by first making a row of finite models and after that constructing the projective limit. We will prove for each finite model that it makes ACP_u, *RDP*, *RSP*, *AIP*, *EA*, *ODP* and *OSP* true. Then we will prove the same items for the projective limit, using the results for the finite models. In fact, we show that the properties that are valid for the finite models are preserved under projective limits. This is a subject of research in model theory and is known under the name of "preservation theorems". In [**19**] we find in exercise 5.2.25* a preservation theorem concerning single-sorted projective limits. It says that a sentence φ is preserved under inverse limits (= projective limits) if and only if φ is equivalent to a sentence of the form

$$\bigwedge_{i=1}^{n} (\forall x_1, \ldots, x_s)\big((\forall y_1, \ldots, y_t)\psi_i \rightarrow \theta_i\big),$$

where ψ_i and θ_i are quantifier-free positive formulas. We will work out every proof since we have two-sorted algebras, but it is the the author's opinion, that it is worthwhile investigating many-sorted inverse (and direct) limits separately in connection to preservation theorems such as the one mentioned above. A general reference to many-sorted algebras is [**35**].

* This is not a note: the asterisk belongs to the name of the exercise.

A1	$x + y = y + x$	$\partial'_H(a) = a, \quad \text{if } a \notin H$	D1′
A2	$x + (y + z) = (x + y) + z$	$\partial'_H(a) = \delta, \quad \text{if } a \in H$	D2′
A3	$x + x = x$	$\partial'_H(x \cdot y) = \partial'_H(x) \cdot \partial'_H(y)$	D3′
A4	$(x + y) \cdot z = x \cdot z + y \cdot z$	$\partial'_H(x + y) = \partial'_H(x) + \partial'_H(y)$	D4′
A5	$(x \cdot y) \cdot z = x \cdot (y \cdot z)$		
A6	$x + \delta = x$	$\pi'_k(a) = a$	PR1′
A7	$\delta \cdot x = \delta$	$\pi'_1(a \cdot x) = a$	PR2′
		$\pi'_{k+1}(a \cdot x) = a \cdot \pi'_k(x)$	PR3′
CM1	$x \parallel y = x \mathbin{\lfloor\!\!\lfloor} y + y \mathbin{\lfloor\!\!\lfloor} x + x \mid y$	$\pi'_k(x + y) = \pi'_k(x) + \pi'_k(y)$	PR4′
CM2	$a \mathbin{\lfloor\!\!\lfloor} x = a \cdot x$		
CM3	$(a \cdot x) \mathbin{\lfloor\!\!\lfloor} y = a \cdot (x \parallel y)$	$a \mid b = b \mid a$	C1
CM4	$(x + y) \mathbin{\lfloor\!\!\lfloor} z = x \mathbin{\lfloor\!\!\lfloor} z + y \mathbin{\lfloor\!\!\lfloor} z$	$(a \mid b) \mid c = a \mid (b \mid c)$	C2
CM5	$(a \cdot x) \mid b = (a \mid b) \cdot x$	$\delta \mid a = \delta$	C3
CM6	$a \mid (b \cdot x) = (a \mid b) \cdot x$		
CM7	$(a \cdot x) \mid (b \cdot y) = (a \mid b) \cdot (x \parallel y)$		
CM8	$(x + y) \mid z = x \mid z + y \mid z$		
CM9	$x \mid (y + z) = x \mid y + x \mid z$		

Table 3.4. An axiom system abbreviated by T.

Definition (3.5.1)

Let G be the set of all closed terms over the theory T (see table 3.4). Let p be an element of G. We define the following subset of G.

$$[p]_n = \{\, q \in G : T \vdash \pi'_n(q) = \pi'_n(p) \,\}.$$

Now we define the set A_n to be the following.

$$A_n = \{\, [p]_n : p \in G \,\}.$$

Definition (3.5.2)

Let $\phi : A_n \longrightarrow A_n$ be a function. Suppose that we have for this function $\phi([\delta]_n) = [\delta]_n$, $\phi(x + y) = \phi(x) + \phi(y)$ for all $x, y \in A_n$ and for all $p \in G$ and l with $1 \le l \le n$ we have

$$\phi\big([p]_n\big) = [q]_n \Longrightarrow \phi\big(\big[\pi'_l(p)\big]_n\big) = \big[\pi'_l(q)\big]_n,$$

then we will call such a function a *laminal* function. We will use the abbreviation H_n for the set of all laminal functions.

Definition (3.5.3)

Let $p, q \in G$. We will define here some operators. First the binary operators, with both arguments in A_n.

$$[p]_n \star [q]_n = [p \star q]_n, \text{ for } \star = +, \cdot, \|, \,\rule{0pt}{0pt}\!\lfloor\!\lfloor, | \,.$$

Secondly the unary operators with their argument in A_n. Let $k \geq 1$. Then we define the projection operator $\pi_k : A_n \longrightarrow A_n$ to be $\pi_k\big([p]_n\big) = \big[\pi'_k(p)\big]_n$. Now let H be a subset of the set of atomic actions, then we define the encapsulation operator $\partial_H : A_n \longrightarrow A_n$ to be $\partial_H\big([p]_n\big) = \big[\partial'_H(p)\big]_n$.

Lemma (3.5.4)

Let $x, y \in G$. Let $k, l \geq 1$. Let $H \subseteq A$. Then the following holds.

(i) $\quad \pi'_k \circ \pi'_l(x) = \pi'_{\min(k,l)}(x)$

(ii) $\quad \pi'_k \circ \partial'_H(x) = \partial'_H \circ \pi'_k(x)$

(iii) $\quad \pi'_k(x \cdot y) = \pi'_k\big(\pi'_k(x) \cdot \pi'_k(y)\big)$

(iv) $\quad \pi'_k(x \mid y) = \pi'_k\big(\pi'_k(x) \mid \pi'_k(y)\big)$

(v) $\quad \pi'_k(x \,\lfloor\!\lfloor\, y) = \pi'_k\big(\pi'_k(x) \,\lfloor\!\lfloor\, \pi'_k(y)\big)$

(vi) $\quad \pi'_k(x \parallel y) = \pi'_k\big(\pi'_k(x) \parallel \pi'_k(y)\big)$

Proof. Most of these properties have been proved in chapter 2 for processes that can be defined with the aid of a guarded recursive specification; see section 2.4. The proof of (ii) is completely analogous to the proof of theorem (3.4.8). This will end the proof of lemma 3.5.4.

Lemma (3.5.5)

The operations that we introduced in (3.5.3) are well-defined, i.e., the operators are independent of the choice of the representatives.

Proof. Suppose that $p' \in [p]_n$ and $q' \in [q]_n$. Then we obviously have:

$$\pi'_n(p') = \pi'_n(p), \text{ and } \pi'_n(q') = \pi'_n(q).$$

Consider the following calculation:

$$\begin{aligned}\pi'_n(p' + q') &= \pi'_n(p') + \pi'_n(q') \\ &= \pi'_n(p) + \pi'_n(q) \\ &= \pi'_n(p + q).\end{aligned}$$

Thus, we find that $[p' + q']_n = [p + q]_n$ and the alternative composition is independent of the choice of the representatives. Now let $\star = \cdot, \|, \lfloor\!\lfloor$ or $|$. Then consider the following:

$$\begin{aligned}\pi'_n(p' \star q') &= \pi'_n\big(\pi'_n(p') \star \pi'_n(q')\big) \\ &= \pi'_n\big(\pi'_n(p) \star \pi'_n(q)\big) \\ &= \pi'_n(p \star q).\end{aligned}$$

Here, we make use of lemma (3.5.4). We will consider the unary operators that we introduced in (3.5.3). Let $p' \in [p]_n$. Then we have for $k \geq 1$:

$$\begin{aligned}\pi'_n \circ \pi'_k(p') &= \pi'_{\min(k,n)}(p')\\ &= \pi'_k \circ \pi'_n(p')\\ &= \pi'_k \circ \pi'_n(p)\\ &= \pi'_{\min(k,n)}(p)\\ &= \pi'_n \circ \pi'_k(p).\end{aligned}$$

Thus, we find $T \vdash \pi'_n \circ \pi'_k(p') = \pi'_n \circ \pi'_k(p)$. Let $H \subseteq A$ and $p' \in [p]_n$. Then, we have the following.

$$\begin{aligned}\pi'_n \circ \partial'_H(p') &= \partial'_H \circ \pi'_n(p')\\ &= \partial'_H \circ \pi'_n(p)\\ &= \pi'_n \circ \partial'_H(p).\end{aligned}$$

So we see that $T \vdash \pi'_n \circ \partial'_H(p') = \pi'_n \circ \partial'_H(p)$; thus we find that $\partial_H\big([p]_n\big) = \partial_H\big([p']_n\big)$. And we see that all the operators that we introduced in (3.5.3) are well-defined. This ends the proof of our lemma.

Lemma (3.5.6)

Let $n \geq 1$. For all $k \geq 1$ we have $\pi_k \in H_n$. For all $H \subseteq A$ we have $\partial_H \in H_n$.

Proof. We are to show for these operators that they are laminal functions. For the definition of a laminal function, see (3.5.2). Let $n, k \geq 1$. It is evidently clear, that π_k is the identity on $[\delta]_n$. The same is valid for ∂_H, for every subset $H \subseteq A$. Let $[p]_n, [q]_n \in A_n$ and let ϕ be the projection operator, or the encapsulation operator; and let ϕ' be the acuted corresponding operator. Then consider the following.

$$\begin{aligned}\phi\big([p]_n + [q]_n\big) &= \phi\big([p+q]_n\big)\\ &= \big[\phi'(p+q)\big]_n\\ &= \big[\phi'(p)\big]_n + \big[\phi'(q)\big]_n\\ &= \phi\big([p]_n\big) + \phi\big([q]_n\big).\end{aligned}$$

It follows immediately from lemma (3.5.4)(i) that π_k is a laminal function. It follows at once from lemma (3.5.4)(ii) that ∂_H is a laminal function. This will end the proof of lemma 3.5.6.

Remark (3.5.7)

We know that $|H_n| < \infty$, so $\{\pi_k : k \geq 1\}$ must be finite. Let $[p]_n \in A_n$. Because of lemma (3.5.4), we can find the following for all $k \geq 0$:

$$\pi'_{n+k} \circ \pi'_n(p) = \pi'_n(p).$$

Thus, we find $\pi_{n+k}\big([p]_n\big) = \pi_n\big([p]_n\big)$ for all $k \geq 0$ and $[p]_n \in A_n$. Moreover, we see that π_{n+k} is the identity map for all $k \geq 0$.

Definition (3.5.8)

We define $\chi : H_n \times A_n \longrightarrow A_n$ as follows: $\chi(f,x) = f(x)$, for $f \in H_n$ and $x \in A_n$. We define the composition of functions $\circ : H_n \times H_n \longrightarrow H_n$ as follows. Let $(f,g) \in H_n \times H_n$, then $f \circ g : A_n \longrightarrow A_n$ is defined: $f \circ g(x) = f\big(g(x)\big)$, for $x \in A_n$. It is easy to see that $f \circ g$ is indeed a laminal function.

Now we are in a position to give the following definition.

Definition (3.5.9)

Let $\mathfrak{M}_n$ be the algebra that consists of the sets A_n and H_n, the operators $+, \cdot, \|, \mathbin{|\!\!\lfloor}, |, \chi, \circ$ and the constants $[a]_n \in A_n$ for all $a \in A$ and the constants $\pi_k, \partial_H \in H_n$ for every $k \geq 1$ and $H \subseteq A$.

A1	$x + y = y + x$	$\partial_H(a) = a, \quad \text{if } a \notin H$	D1
A2	$x + (y + z) = (x + y) + z$	$\partial_H(a) = \delta, \quad \text{if } a \in H$	D2
A3	$x + x = x$	$\partial_H(x \cdot y) = \partial_H(x) \cdot \partial_H(y)$	D3
A4	$(x + y) \cdot z = x \cdot z + y \cdot z$		
A5	$(x \cdot y) \cdot z = x \cdot (y \cdot z)$	$\pi_n(a) = a$	PR1
A6	$x + \delta = x$	$\pi_1(a \cdot x) = a$	PR2
A7	$\delta \cdot x = \delta$	$\pi_{n+1}(a \cdot x) = a \cdot \pi_n(x)$	PR3
CM1	$x \parallel y = x \mathbin{\lfloor\!\lfloor} y + y \mathbin{\lfloor\!\lfloor} x + x \mid y$	$a \mid b = b \mid a$	C1
CM2	$a \mathbin{\lfloor\!\lfloor} x = a \cdot x$	$(a \mid b) \mid c = a \mid (b \mid c)$	C2
CM3	$(a \cdot x) \mathbin{\lfloor\!\lfloor} y = a \cdot (x \parallel y)$	$\delta \mid a = \delta$	C3
CM4	$(x + y) \mathbin{\lfloor\!\lfloor} z = x \mathbin{\lfloor\!\lfloor} z + y \mathbin{\lfloor\!\lfloor} z$		
CM5	$(a \cdot x) \mid b = (a \mid b) \cdot x$	$\chi(f \circ g, x) = \chi\big(f, \chi(g,x)\big)$	XC1
CM6	$a \mid (b \cdot x) = (a \mid b) \cdot x$	$\chi\big((f \circ g) \circ h, x\big) = \chi\big(f \circ (g \circ h), x\big)$	XC2
CM7	$(a \cdot x) \mid (b \cdot y) = (a \mid b) \cdot (x \parallel y)$	$\chi(f, \gamma) = \gamma$	X1
CM8	$(x + y) \mid z = x \mid z + y \mid z$	$\chi(f, \gamma \cdot x) = \gamma \cdot \chi(f,x)$	X2
CM9	$x \mid (y + z) = x \mid y + x \mid z$	$\chi(f, x + y) = \chi(f,x) + \chi(f,y)$	X3

Table 3.5. ACP_u.

Theorem (3.5.10)

Let $n \geq 1$, then we have $\mathfrak{M}_n \models ACP_u$. See table 3.5 on page 115 for the axiom system ACP_u.

Proof. We are to show that $\mathfrak{M}_n$ models each axiom of ACP_u. We will only sketch the proof, since the other axioms are deduced in the same way as the examples below are proved. We will show that $\mathfrak{M}_n \models$ A4.

$$\begin{aligned}\big([p]_n + [q]_n\big) \cdot [r]_n &= \big([p+q]_n\big) \cdot [r]_n\\ &= \big[(p+q)\cdot r\big]_n\\ &= [p\cdot r + q\cdot r]_n\\ &= [p\cdot r]_n + [q\cdot r]_n\\ &= [p]_n\cdot[r]_n + [q]_n\cdot[r]_n.\end{aligned}$$

Now we will infer $\mathfrak{M}_n \models$ CM7.

$$\begin{aligned}\big([a]_n\cdot[p]_n\big) \mid \big([b]_n\cdot[q]_n\big) &= [a\cdot p]_n \mid [b\cdot q]_n\\ &= \big[(a\cdot p)\mid(b\cdot q)\big]_n\\ &= \big[(a\mid b)\cdot(p\parallel q)\big]_n\\ &= [a\mid b]_n\cdot[p\parallel q]_n\\ &= \big([a]_n \mid [b]_n\big)\cdot\big([p]_n \parallel [q]_n\big).\end{aligned}$$

We will prove $\mathfrak{M}_n \models$ D3.

$$\begin{aligned}\chi\big(\partial_H, [p]_n + [q]_n\big) &= \chi\big(\partial_H, [p+q]_n\big)\\ &= \partial_H\big([p+q]_n\big)\\ &= \big[\partial'_H(p+q)\big]_n\\ &= \big[\partial'_H(p)+\partial'_H(q)\big]_n\\ &= \big[\partial'_H(p)\big]_n + \big[\partial'_H(q)\big]_n\\ &= \partial_H\big([p]_n\big) + \partial_H\big([q]_n\big)\\ &= \chi\big(\partial_H,[p]_n\big) + \chi\big(\partial_H,[q]_n\big).\end{aligned}$$

We will show $\mathfrak{M}_n \models$ PR3. Let $k \geq 1$.

$$\begin{aligned}\chi\big(\pi_{k+1}, [a]_n\cdot[p]_n\big) &= \pi_{k+1}\big([a\cdot p]_n\big)\\ &= \big[\pi'_{k+1}(a\cdot p)\big]_n\\ &= \big[a\cdot\pi'_k(p)\big]_n\\ &= [a]_n\cdot\big[\pi'_k(p)\big]_n\\ &= [a]_n\cdot\pi_k\big([p]_n\big)\\ &= [a]_n\cdot\chi\big(\pi_k,[p]_n\big).\end{aligned}$$

Now we will prove XC1 and X2. Suppose that $h, k \in H_n$ and $x \in A_n$.

$$\begin{aligned}\chi(h\circ k, x) &= h\circ k(x)\\ &= \chi\big(h, k(x)\big)\\ &= \chi\big(h,\chi(k,x)\big).\end{aligned}$$

Let $h \in H_n$ and let $[p]_n \in A_n$.

$$\begin{aligned}
\chi\big(h, [\delta]_n \cdot [p]_n\big) &= h\big([\delta]_n \cdot [p]_n\big) \\
&= h\big([\delta]_n\big) \\
&= [\delta]_n \\
&= \big[\delta \cdot \chi\big(h, [p]_n\big)\big]_n \\
&= [\delta]_n \cdot \chi\big(h, [p]_n\big).
\end{aligned}$$

This ends the sketch of the proof of 3.5.10.

RA2	$(x+y)+z \to x+(y+z)$	$\partial'_H(a) \to a, \quad \text{if } a \notin H$	RD1′
RA3	$x+x \to x$	$\partial'_H(a) \to \delta, \quad \text{if } a \in H$	RD2′
RA4	$(x+y)\cdot z \to x\cdot z + y\cdot z$	$\partial'_H(x\cdot y) \to \partial'_H(x)\cdot\partial'_H(y)$	RD3′
RA5	$(x\cdot y)\cdot z \to x\cdot(y\cdot z)$	$\partial'_H(x+y) \to \partial'_H(x)+\partial'_H(y)$	RD4′
RA6	$x+\delta \to x$		
RA7	$\delta\cdot x \to \delta$	$\pi'_k(a) \to a$	RPR1′
		$\pi'_1(a\cdot x) \to a$	RPR2′
RCM1	$x \parallel y \to x \mathbin{⌊\!⌊} y + (y \mathbin{⌊\!⌊} x + x \mid y)$	$\pi'_{k+1}(a\cdot x) \to a\cdot\pi'_k(x)$	RPR3′
RCM2	$a \mathbin{⌊\!⌊} x \to a\cdot x$	$\pi'_k(x+y) \to \pi'_k(x)+\pi'_k(y)$	RPR4′
RCM3	$(a\cdot x) \mathbin{⌊\!⌊} y \to a\cdot(x \parallel y)$		
RCM4	$(x+y) \mathbin{⌊\!⌊} z \to x \mathbin{⌊\!⌊} z + y \mathbin{⌊\!⌊} z$	$a \mid b \to c_{a,b}$	RC
RCM5	$(a\cdot x) \mid b \to (a \mid b)\cdot x$		
RCM6	$a \mid (b\cdot x) \to (a \mid b)\cdot x$		
RCM7	$(a\cdot x) \mid (b\cdot y) \to (a \mid b)\cdot(x \parallel y)$		
RCM8	$(x+y) \mid z \to x \mid z + y \mid z$		
RCM9	$x \mid (y+z) \to x \mid y + x \mid z$		

Table 3.6. A term rewriting system associated with the theory T.

Lemma (3.5.11)

The term rewriting system of table 3.6 has the termination property*.

* See definition (3.3.11)

Proof. We will prove 3.5.11 in the same way as we proved (3.3.14). Firstly, we will give the partial ordering of the signature. We will also use the ranked operators that we introduced in section 3.3.

$$\|_n > \mathbin{|\!\!\lfloor}_n, |_n > \|_{n-1}, \quad \|_n, \mathbin{|\!\!\lfloor}_n, |_n > \cdot, \quad \partial_H, \cdot > +, \quad \pi_{n+1} > \pi_n > \cdot, \quad |_2 > A > \delta.$$

We will use the lexicographical variant of the recursive path ordering, although we will need the lexicographical status only for the alternative composition and for the sequential composition. We are to show that for each rewriting rule $s \to t$ in table 3.6, $s \succ t$. For the symbol "$\succ$" we refer to definition (3.3.9). The treatment of the cases RA2–RA7, RCM1–RCM9 and RC, is the same as in theorem (3.3.14). The calculation of RPR1′ is trivial. Let us treat RPR2′.

$$\begin{aligned}
\pi'_1(a \cdot x) &\succ \pi'^*_1(a \cdot x) \\
&\succ a \cdot x \\
&\succ a \cdot^* x \\
&\succ a.
\end{aligned}$$

Now we will handle RPR3′. Let $k \geq 1$.

$$\begin{aligned}
\pi'_{k+1}(a \cdot x) &\succ \pi'^*_{k+1}(a \cdot x) \\
&\succ \pi'^*_{k+1}(a \cdot x) \cdot \pi'^*_{k+1}(a \cdot x) \\
&\succ (a \cdot x) \cdot \pi'_k\big(\pi'^*_{k+1}(a \cdot x)\big) \\
&\succ (a \cdot^* x) \cdot \pi'_k(a \cdot x) \\
&\succ a \cdot \pi'^*_k(a \cdot x) \\
&\succ a \cdot \pi'_k(a \cdot^* x) \\
&\succ a \cdot \pi'_k(x).
\end{aligned}$$

Now we will discuss RPR4′. Let $k \geq 1$.

$$\begin{aligned}
\pi'_k(x + y) &\succ \pi'^*_k(x + y) \\
&\succ \pi'_k(x + y) +^* \pi'_k(x + y) \\
&\succ \pi'^*_k(x + y) + \pi'^*_k(x + y) \\
&\succ \pi'_k(x +^* y) + \pi'_k(x +^* y) \\
&\succ \pi'_k(x) + \pi'_k(y).
\end{aligned}$$

RD1–RD4 are proved in the same way as in theorem (3.3.14), albeit that we use ∂'_H in the partial ordering instead of χ. This ends the proof of 3.5.11.

Lemma (3.5.12)

Let $n \geq 1$ be chosen. Let $p \in G$ be a closed term over the theory T. See (3.5.1) for the definition of G. Then there is an element $q \in G$, such that $[q]_n = [p]_n$ and $T \vdash \pi'_n(q) = q$. We will call q the n-normal form of p.

Proof. Since $p \in G$ is a closed T-term, $\pi'_n(p)$ is also a closed T-term. With the aid of lemma (3.5.11), we can rewrite this term to a term $q \in G$, which is in normal form. We know that $T \vdash q = \pi'_n(p)$, thus,

$$\begin{aligned} \pi'_n(q) &= \pi'_n\big(\pi'_n(p)\big) \\ &= \pi'_n(p) \\ &= q, \end{aligned}$$

with the aid of lemma (3.5.4) and we also see that $[q]_n = [p]_n$. This ends the proof of 3.5.12.

Lemma (3.5.13)

Let $p', q' \in G$ be closed terms and let p and q be their n-normal forms. Then we have the following.

$$\mathfrak{M}_n \models p = q \iff T \vdash p = q.$$

Proof. Let us assume that $\mathfrak{M}_n \models p = q$, with p and q n-normal forms. Then we find that $T \vdash \pi'_n(p) = \pi'_n(q)$. But because of lemma (3.5.12), we have $\pi'_n(p) = p$ and $\pi'_n(q) = q$. So we find $T \vdash p = q$.

Now let us assume that $T \vdash p = q$. Then we have a fortiori $T \vdash \pi'_n(p) = \pi'_n(q)$. So $[p]_n = [q]_n$ and we find $\mathfrak{M}_n \models p = q$. This ends the proof of 3.5.13.

Epiphenomenon (3.5.14)

Let $a_i = |A_i|$ be the number of elements of the set A_i that we defined in (3.5.1) ($i \geq 1$) and let $a_0 = 0$. Let $u = |A|$. Then we have the following recursive formula.

$$\begin{aligned} a_0 &= 0, \\ a_{i+1} &= 2^{u\cdot(1+a_i)}. \end{aligned}$$

Proof. We will give a sketch of the proof. Let $i = 1$. Because of axioms A3 and A6, we can make the following different sums:

$$1 + \sum_{i=1}^{u} \binom{u}{i} = 1 + (2^u - 1) = 2^u = 2^{u\cdot(1+a_0)}.$$

Let $i > 1$. We will have $u + u \cdot a_i$ terms of which the first symbol (in prefix notation) is not a sum. Thus, we obtain, just as above for $i = 1$, $2^{u\cdot(1+a_i)}$ possibilities for the number of different sums that we can make with these $u \cdot (1 + a_i)$ terms of which the sums are built up.

u	1	2	3	4	5	6
a_1	2	4	8	16	32	64
a_2	8	1024	$1.3 \cdot 10^8$	$2.9 \cdot 10^{20}$	$4.6 \cdot 10^{49}$	$2.5 \cdot 10^{117}$
a_3	512	$1.2 \cdot 10^{617}$	$-$*	$-$	$-$	$-$
a_4	$2.6 \cdot 10^{154}$	$-$	$-$	$-$	$-$	$-$

* ($> 10^{9999}$)

Table 3.7. Some calculations with (3.5.14).

Examples (3.5.15)

In table 3.7 we will give some figures that we have calculated with the aid of the formula of (3.5.14). For instance, if we have two atomic actions, there are approximately $1.2 \cdot 10^{617}$ elements in A_3.

Remark (3.5.16)

Let x be a variable. Let $C[x]$ be a context of x, in which the occurrence of x is guarded. Then we have for all $k \geq n$:

$$\pi'_{n+1}\big(C[x]\big) = \pi'_{n+1}\big(C\big[\pi'_k(x)\big]\big).$$

Theorem (3.5.17)

Let $n \geq 1$, then we have $\mathfrak{M}_n \models RDP, RSP$.

Proof. Let $V = \{x_\alpha : \alpha \in I\}$ be a set of variables. Let

$$E(V) = \big\{\, x_\alpha = t_\alpha\big((x_\beta)_{\beta \in I}\big) : \alpha \in I \,\big\}$$

be a guarded recursive specification. Without loss of generality, we may assume that $E(V)$ is completely guarded; see definition (3.2.15). We will prove 3.5.17 with induction on n. Thus, let $n = 1$. For all $\alpha \in I$, we calculate $p^1_\alpha := \pi'_1(x_\alpha)$. This is a closed term over the theory T.

$$\begin{aligned} \pi'_1\big(t_\alpha\big((p^1_\beta)_\beta\big)\big) &= \pi'_1\big(t_\alpha\big((x_\beta)_\beta\big)\big) && (1)\\ &= \pi'_1(x_\alpha)\\ &= \pi'_1 \circ \pi'_1(x_\alpha)\\ &= \pi'_1(p^1_\alpha). \end{aligned}$$

In (1) we use the fact that $\pi'_1(a \cdot x) = \pi'_1(a \cdot y)$ for all x and y in combination with the fact that $E(V)$ is completely guarded. Henceforward, we will use this argument tacitly. So we see that $(p^1_\alpha)_\alpha$ solves the system $E(V)$ in $\mathfrak{M}_1$. Now suppose that $(q_\alpha)_\alpha$ also solves this system, i.e., $\pi'_1(q_\alpha) = \pi'_1\big(t_\alpha\big((q_\beta)_\beta\big)\big)$. Then we find the following:

$$\begin{aligned}\pi'_1(q_\alpha) &= \pi'_1\big(t_\alpha\big((q_\beta)_\beta\big)\big)\\ &= \pi'_1\big(t_\alpha\big((x_\beta)_\beta\big)\big)\\ &= \pi'_1(x_\alpha)\\ &= \pi'_1 \circ \pi'_1(x_\alpha)\\ &= \pi'_1(p^1_\alpha).\end{aligned}$$

We see that the solution $(q_\alpha)_\alpha$ is just the one that we have already constructed, so $(p^1_\alpha)_\alpha$ is the unique solution. Now we see that 3.5.17 is correct for $n = 1$. Let $n \geq 1$ and suppose that 3.5.17 has been proved for n. Let $(p^n_\alpha)_\alpha$ be such that

$$T \vdash \pi'_n(p^n_\alpha) = \pi'_n\big(t_\alpha\big((p^n_\beta)_\beta\big)\big).$$

Define $p^{n+1}_\alpha := t_\alpha\big((p^n_\beta)_\beta\big)$. We will show that

$$\pi'_{n+1}(p^{n+1}_\alpha) = \pi'_{n+1}\big(t_\alpha\big((p^{n+1}_\beta)_\beta\big)\big).$$

Consider thereto the following.

$$\begin{aligned}\pi'_{n+1}(p^{n+1}_\alpha) &= \pi'_{n+1}\big(t_\alpha\big((p^n_\beta)_\beta\big)\big)\\ &= \pi'_{n+1}\Big(t_\alpha\big(\big(\pi'_n(p^n_\beta)\big)_\beta\big)\Big) \qquad (2)\\ &= \pi'_{n+1}\Big(t_\alpha\big(\pi'_n\big(t_\beta\big((p^n_\gamma)_\gamma\big)\big)_\beta\big)\Big)\\ &= \pi'_{n+1}\Big(t_\alpha\big(t_\beta\big((p^n_\gamma)_\gamma\big)_\beta\big)\Big)\\ &= \pi'_{n+1}\big(t_\alpha\big((p^{n+1}_\beta)_\beta\big)\big).\end{aligned}$$

In equation (2), we use the fact that $E(V)$ is completely guarded and we use remark (3.5.16). In the sequel of this proof, we will apply this argument tacitly. We find that $(p^{n+1}_\alpha)_\alpha$ solves $E(V)$ in $\mathfrak{M}_{n+1}$. Now we will show that this solution is unique. Let $(q_\alpha)_\alpha$ be such that

$$T \vdash \pi'_{n+1}(q_\alpha) = \pi'_{n+1}\big(t_\alpha\big((q_\beta)_\beta\big)\big).$$

Observe that the following holds.

$$\begin{aligned}\pi'_n(p^{n+1}_\alpha) &= \pi'_n\big(t_\alpha\big((p^n_\beta)_\beta\big)\big)\\ &= \pi'_n(p^n_\alpha). \qquad (3)\end{aligned}$$

First, we will show that $\pi'_n(q_\alpha)$ is a solution in $\mathfrak{M}_n$.

$$\begin{aligned}\pi'_n(q_\alpha) &= \pi'_n \circ \pi'_{n+1}(q_\alpha)\\ &= \pi'_n \circ \pi'_{n+1}\big(t_\alpha\big((q_\beta)_\beta\big)\big)\\ &= \pi'_n\big(t_\alpha\big((q_\beta)_\beta\big)\big).\end{aligned}$$

But in $\mathfrak{M}_n$ the solution is unique, so we find with (3) that

$$\pi'_n(q_\alpha) = \pi'_n(p_\alpha) = \pi'_n(p_\alpha^{n+1}).$$

Now we will show that $\pi'_{n+1}(q_\alpha) = \pi'_{n+1}(p_\alpha^{n+1})$.

$$\begin{aligned}
\pi'_{n+1}(q_\alpha) &= \pi'_{n+1}\big(t_\alpha\big((q_\beta)_\beta\big)\big) \\
&= \pi'_{n+1}\Big(t_\alpha\big(\big(\pi'_n(q_\beta)\big)_\beta\big)\Big) \\
&= \pi'_{n+1}\Big(t_\alpha\big(\big(\pi'_n(p_\beta^{n+1})\big)_\beta\big)\Big) \\
&= \pi'_{n+1}\big(t_\alpha\big((p_\beta^{n+1})_\beta\big)\big) \\
&= \pi'_{n+1}(p_\alpha^{n+1}).
\end{aligned}$$

This ends the proof of the theorem.

Theorem (3.5.18)

Let $n \geq 1$, then $\mathfrak{M}_n \models AIP$.

Proof. Let $x, y \in A_n$. Suppose that for all $k \geq 1$, we have $\pi_k(x) = \pi_k(y)$. In particular we have this for $k = n$. With the aid of remark (3.5.7), we find that $\pi_n = id$, so we see that $x = y$. This will end the proof of 3.5.18.

Lemma (3.5.19)

Let $[p]_n \in A_n$. There are $a_1, \ldots, a_s, b_1, \ldots, b_t \in A \cup \{\delta\}$ $p_1, \ldots, p_s \in G$, such that

$$[p]_n = \sum_{j=1}^{s} [a_j]_n \cdot [p_j]_n + \sum_{k=1}^{t} [b_k]_n.$$

Proof. Let p be a closed term over the theory T. See table 3.4 on page 112. If we take a close look at this axiom system, we see that this is in fact ACP with projections. From this system we know that there are unique normal forms for all closed terms (see [**8**]). They have the desired form:

$$p = \sum_{k=1}^{s} a_j \cdot p_j + \sum_{k=1}^{t} b_k,$$

for certain $a_1, \ldots, a_s, b_1, \ldots, b_t \in A \cup \{\delta\}$ and closed terms $p_1, \ldots, p_s$. Thus, we find

$$[p]_n = \sum_{j=1}^{s} [a_j]_n \cdot [p_j]_n + \sum_{k=1}^{t} [b_k]_n.$$

This ends the proof of the lemma.

Theorem (3.5.20)

Let $n \geq 1$, then $\mathfrak{M}_n \models EA$. See (4.2.1) for the definition of EA.

Proof. Since H_n is defined as a function space, we automatically have that for all $f, g \in H_n : f = g$, if and only if we have for all $x \in A_n : f(x) = g(x)$. This is what we wanted to prove.

Theorem (3.5.21)

Let $n \geq 1$, then $\mathfrak{M}_n \models ODP, OSP$.

Proof. Fix an $n \geq 1$. We will use the more explicit formulation of the definition of a linear functional specification that we have already described in (3.2.3). Let $M = \{m_1, \ldots, m_l\}$ be a set of function names and let

$$\sigma : A \times \{1, \ldots, l\} \longrightarrow \{1, \ldots, l\}$$

be a given map. Recall that the linear functional specification $E(M)$ has the following form:

$$\begin{aligned} E(M) = \big\{\, & m_i(a) = a^{(i)} : a \in A,\, 1 \leq i \leq l \,\big\} \\ & \cup \big\{\, m_i(a \cdot x) = a^{(i)} \cdot m_{\sigma(a,i)}(x) : a \in A,\, 1 \leq i \leq l \,\big\}. \end{aligned}$$

First we will prove that there is a valuation $\varphi : M \longrightarrow H_n$, which solves the system of equations $E(M)$. See section 3.2 for the definition of a valuation. It suffices to show that there are $\mu_1, \ldots, \mu_l \in H_n$ such that

$$\mu_i\big([a]_n\big) = \big[a^{(i)}\big]_n, \tag{4}$$

$$\mu_i\big([a]_n \cdot [p]_n\big) = \big[a^{(i)}\big]_n \cdot \mu_{\sigma(a,i)}\big([p]_n\big). \tag{5}$$

(So we can take $\varphi(m_i) := \mu_i$.) Let $[p]_n \in A_n$, there are $a_1, \ldots, a_s, b_1, \ldots, b_t$ in $A \cup \{\delta\}$ and $p_1, \ldots, p_s \in G$ such that

$$[p]_n = \sum_{j=1}^{s} [a_j]_n \cdot [p_j]_n + \sum_{k=1}^{t} [b_k]_n, \tag{6}$$

according to lemma (3.5.19). Let $\mu_i^1 : A_n \longrightarrow A_n$ be defined as follows:

$$\begin{aligned} \mu_i^1\big([\delta]_n\big) &= [\delta]_n, \\ \mu_i^1\big([p]_n\big) &= \sum_{j=1}^{s} \big[a_j^{(i)}\big]_n + \sum_{k=1}^{t} \big[b_k^{(i)}\big]_n. \end{aligned}$$

It consists of a straightforward calculation to show that $\mu_i^1 \in H_n$. Let $1 \leq r < n$. Suppose that $\mu_i^r \in H_n$, for all $1 \leq i \leq l$, then we define

$$\mu_i^{r+1} : A_n \longrightarrow A_n$$

to be the identity on $[\delta]_n$ and for $[p]_n \in A_n$

$$\mu_i^{r+1}\big([p]_n\big) = \sum_{j=1}^{s} \big[a_j^{(i)}\big]_n \cdot \mu_{\sigma(a_j,i)}^{r}\big([p_j]_n\big) + \sum_{k=1}^{t} \big[b_k^{(i)}\big]_n.$$

We will show that $\mu_i^{r+1} \in H_n$. It is trivial to prove that μ_i^{r+1} distributes over the alternative composition. Let $[p]_n \in A_n$ be as in equation (6). Let

$$\begin{aligned} [q_j]_n &= \mu_{\sigma(a_j,i)}^{r}\big([p_j]_n\big) \\ q &= \sum_{j=1}^{s} a_j^{(i)} \cdot q_j + \sum_{k=1}^{t} b_k^{(i)}. \end{aligned}$$

Then we see that $\mu_i^{r+1}\big([p]_n\big) = [q]_n$. We will prove for all v, with $1 \le v \le n$ that

$$\mu_i^{r+1}\big(\big[\pi_v'(p)\big]_n\big) = \big[\pi_v'(q)\big]_n. \tag{7}$$

If $v = 1$, we immediately see that equation (7) is valid. Now suppose that $1 < v \le n$, then consider the subsequent deduction.

$$\begin{aligned} \mu_i^{r+1}\big(\big[\pi_v'(p)\big]_n\big) &= \sum_{j=1}^{s} \big[a_j^{(i)}\big]_n \cdot \mu_{\sigma(a_j,i)}^{r}\big(\big[\pi_{v-1}'(p_j)\big]_n\big) + \sum_{k=1}^{t} \big[b_k^{(i)}\big]_n \\ &= \sum_{j=1}^{s} \big[a_j^{(i)}\big]_n \cdot \big[\pi_{v-1}'(q_j)\big]_n + \sum_{k=1}^{t} \big[b_k^{(i)}\big]_n \\ &= \big[\pi_v'(q)\big]_n. \end{aligned}$$

This shows that μ_i^{r+1} is a laminal function. We will prove a technical result on these laminal functions.

$$\pi_{n-r} \circ \mu_i^{n-r} = \pi_{n-r} \circ \mu_i^{n-r+1}, \quad 1 \le i \le l,\ 1 \le r \le n-1. \tag{8}$$

We will prove (8) first for $r = n-1$. Let $[p]_n \in A_n$ be as in equation (6). Consider the calculation hereinafter.

$$\begin{aligned} \pi_1 \circ \mu_i^{2}\big([p]_n\big) &= \pi_1\Big(\sum_{j=1}^{s} \big[a_j^{(i)}\big]_n \cdot \mu_{\sigma(a_j,i)}^{1}\big([p_j]_n\big) + \sum_{k=1}^{t} \big[b_k^{(i)}\big]_n\Big) \\ &= \sum_{j=1}^{s} \big[a_j^{(i)}\big]_n + \sum_{k=1}^{t} \big[b_k^{(i)}\big]_n \\ &= \pi_1\Big(\sum_{j=1}^{s} \big[a_j^{(i)}\big]_n + \sum_{k=1}^{t} \big[b_k^{(i)}\big]_n\Big) \\ &= \pi_1 \circ \mu_i^{1}\big([p]_n\big). \end{aligned}$$

So for $r = n - 1$ we see that (8) is correct. Suppose that (8) has been verified for $1 < r \leq n-1$. We will prove it for $r-1$. Let $[p]_n \in A_n$ be as in equation (6). Consider the following.

$$\begin{aligned}
\pi_{n-r+1} \circ \mu_i^{n-r+1}\big([p]_n\big) &= \pi_{n-r+1}\Big(\sum_{j=1}^{s} \big[a_j^{(i)}\big]_n \cdot \mu_{\sigma(a_j,i)}^{n-r}\big([p_j]_n\big) + \sum_{k=1}^{t} \big[b_k^{(i)}\big]_n\Big) \\
&= \sum_{j=1}^{s} \big[a_j^{(i)}\big]_n \cdot \pi_{n-r} \circ \mu_{\sigma(a_j,i)}^{n-r}\big([p_j]_n\big) + \sum_{k=1}^{t} \big[b_k^{(i)}\big]_n \\
&= \sum_{j=1}^{s} \big[a_j^{(i)}\big]_n \cdot \pi_{n-r} \circ \mu_{\sigma(a_j,i)}^{n-r+1}\big([p_j]_n\big) + \sum_{k=1}^{t} \big[b_k^{(i)}\big]_n \\
&= \pi_{n-r+1}\Big(\sum_{j=1}^{s} \big[a_j^{(i)}\big]_n \cdot \mu_{\sigma(a_j,i)}^{n-r+1}\big([p_j]_n\big) + \sum_{k=1}^{t} \big[b_k^{(i)}\big]_n\Big) \\
&= \pi_{n-r+1} \circ \mu_i^{n-r+2}\big([p]_n\big).
\end{aligned}$$

This will end the verification of (8). In particular, we find that (8) is valid for $r = 1$. Thus, we obtain the following formula:

$$\pi_{n-1} \circ \mu_i^{n-1} = \pi_{n-1} \circ \mu_i^{n}. \tag{9}$$

Define $\mu_i = \mu_i^n \in H_n$ for $1 \leq i \leq l$. We claim that these functions satisfy equations (4) and (5). It will be clear that

$$\mu_i\big([a]_n\big) = \big[a^{(i)}\big]_n,$$

for all $1 \leq i \leq l$, so we see that $\mu_1, \ldots, \mu_l$ satisfy the boundary conditions of $E(M)$, i.e., equation (4). Now let us take a look at the functional equations of $E(M)$, or equivalently, equation (5). Let $i \in \{1, \ldots, l\}$ be fixed.

$$\begin{aligned}
\mu_i\big([a]_n \cdot [p]_n\big) &= \mu_i^n\big([a]_n \cdot [p]_n\big) \\
&= \pi_n \circ \mu_i^n\big([a]_n \cdot [p]_n\big) && \text{see remark (3.5.7)} \\
&= \pi_n\Big(\big[a^{(i)}\big]_n \cdot \mu_{\sigma(a,i)}^{n-1}\big([p]_n\big)\Big) \\
&= \big[a^{(i)}\big]_n \cdot \pi_{n-1} \circ \mu_{\sigma(a,i)}^{n-1}\big([p]_n\big) \\
&= \big[a^{(i)}\big]_n \cdot \pi_{n-1} \circ \mu_{\sigma(a,i)}^{n}\big([p]_n\big) && \text{because of (9)} \\
&= \pi_n\Big(\big[a^{(i)}\big]_n \cdot \mu_{\sigma(a,i)}^{n}\big([p]_n\big)\Big) \\
&= \big[a^{(i)}\big]_n \cdot \mu_{\sigma(a,i)}\big([p]_n\big).
\end{aligned}$$

This will end the proof of the existential part of 3.5.21. Now we will prove the uniqueness part: $\mathfrak{M}_n \models OSP$. Suppose that there is also a valuation

$\phi : M \longrightarrow H_n$, which solves the system of equations $E(M)$. Let $\phi(m_i) := \nu_i$ for all i. Then we have for all $i \in \{1, \ldots, l\}$

$$\begin{aligned} \nu_i\big([a]_n\big) &= \big[a^{(i)}\big]_n, \\ \nu_i\big([a]_n \cdot [p]_n\big) &= \big[a^{(i)}\big]_n \cdot \nu_{\sigma(a,i)}\big([p]_n\big). \end{aligned}$$

In order to verify that $\mu_i = \nu_i$ (for all i), we will prove the following:

$$\pi_r \circ \nu_i = \mu_i^r, \quad 1 \le r \le n,\, 1 \le i \le l. \tag{10}$$

We will prove (10) with induction on r. Let $[p]_n \in A_n$ be as in equation (6). Let $r = 1$ and let $i \in \{1, \ldots, l\}$, then we see the following

$$\begin{aligned} \pi_1 \circ \nu_i\big([p]_n\big) &= \pi_1\Big(\sum_{j=1}^{s} \big[a_j^{(i)}\big]_n \cdot \nu_{\sigma(a_j,i)}\big([p_j]_n\big) + \sum_{k=1}^{t} \big[b_k^{(i)}\big]_n\Big) \\ &= \sum_{j=1}^{s} \big[a_j^{(i)}\big]_n + \sum_{k=1}^{t} \big[b_k^{(i)}\big]_n \\ &= \mu_i^1\big([p]_n\big). \end{aligned}$$

Thus, (10) is correct for $r = 1$. Now suppose that (10) is valid for $1 \le r < n$, then we prove it for $r + 1$. Let $[p]_n \in A_n$ be as in equation (6).

$$\begin{aligned} \pi_{r+1} \circ \nu_i\big([p]_n\big) &= \sum_{j=1}^{s} \big[a_j^{(i)}\big]_n \cdot \pi_r \circ \nu_{\sigma(a_j,i)}\big([p_j]_n\big) + \sum_{k=1}^{t} \big[b_k^{(i)}\big]_n \\ &= \sum_{j=1}^{s} \big[a_j^{(i)}\big]_n \cdot \mu^r_{\sigma(a_j,i)}\big([p_j]_n\big) + \sum_{k=1}^{t} \big[b_k^{(i)}\big]_n \\ &= \mu_i^{r+1}\big([p]_n\big). \end{aligned}$$

This proves (10). In particular we find that (10) is valid for $r = n$. If we combine this with the fact that π_n is the identity map—see remark (3.5.7)—we find for all $1 \le i \le l$ that $\nu_i = \pi_n \circ \nu_i = \mu_i^n = \mu_i$. This will end the verification of the uniqueness part and therewith the proof of 3.5.21.

Construction (3.5.22)

We will construct the projective limit of the models $\mathfrak{M}_n$. Thereto we need to define mappings $\kappa_n : \mathfrak{M}_{n+1} \longrightarrow \mathfrak{M}_n$, for all $n \ge 1$ as follows. First we will define

$$\kappa_n : A_{n+1} \longrightarrow A_n.$$

Let $[p]_{n+1} \in A_{n+1}$, then we define $\kappa_n\big([p]_{n+1}\big) = [p]_n$. Now we will define

$$\kappa_n : H_{n+1} \longrightarrow H_n.$$

Let f be an element of H_{n+1}. We define

$$\kappa_n(f) : A_n \longrightarrow A_n$$

to be the following map. Let $[p]_n \in A_n$, then $\kappa_n(f)\big([p]_n\big) = \kappa_n \circ f\big(\big[\pi'_n(p)\big]_{n+1}\big)$. It is obvious that $\kappa_n(f)$ is well-defined. So let us verify that $\kappa_n(f)$ is an element of H_n. First we are to show that $\kappa_n(f)$ is a laminal function. It is evident that $\kappa_n\big([\delta]_n\big) = [\delta]_n$. To verify that $\kappa_n(f)$ distributes over the alternative composition, we will need the fact that κ_n distributes over the alternative composition. Both proofs are trivial. The following remains to be shown. Let $[p]_n$ be in A_n. Suppose that $f\big(\big[\pi'_n(p)\big]_{n+1}\big) = [q]_{n+1}$. Then it is clear that $\kappa_n(f)\big([p]_n\big) = [q]_n$. We will prove that for all $1 \leq l \leq n$ we have:

$$\kappa_n(f)\big(\big[\pi'_l(p)\big]_n\big) = \big[\pi'_l(q)\big]_n.$$

Consider the calculation hereinafter.

$$\begin{aligned}
\kappa_n(f)\big(\big[\pi'_l(p)\big]_n\big) &= \kappa_n \circ f\big(\big[\pi'_n \circ \pi'_l(p)\big]_{n+1}\big) \\
&= \kappa_n \circ f\big(\big[\pi'_l \circ \pi'_n(p)\big]_{n+1}\big) \\
&= \kappa_n\big(\big[\pi'_l(q)\big]_{n+1}\big) && \text{(since } f \in H_{n+1}) \\
&= \big[\pi'_l(q)\big]_n.
\end{aligned}$$

So we find that $\kappa_n(f) \in H_n$. Now we will show that $\kappa_n : \mathfrak{M}_{n+1} \longrightarrow \mathfrak{M}_n$ is a homomorphism, that is, it distributes over all the operations. Recall that there are seven operations: the merge, the left-merge, the communication-merge, the alternative composition, the sequential composition, the composition of functions and the application function. Let $\star$ be one of $\|, \lfloor\!\lfloor, |, +$ or $\cdot$. Then we are to show that $\kappa_n(x \star y) = \kappa_n(x) \star \kappa_n(y)$, for all $x, y \in A_{n+1}$. This is trivial. We will show that κ_n distributes over the composition of functions. Let $f, g \in H_{n+1}$. Let $[p]_n$ be in A_n. Let $g\big(\big[\pi'_n(p)\big]_{n+1}\big) = [r]_{n+1}$ and let $f\big([r]_{n+1}\big) = [q]_{n+1}$. Now contemplate the following.

$$\begin{aligned}
\kappa_n(f \circ g)\big([p]_n\big) &= \kappa_n \circ (f \circ g)\big(\big[\pi'_n(p)\big]_{n+1}\big) \\
&= \kappa_n\big(f\big([r]_{n+1}\big)\big) \\
&= [q]_n \\
&= \kappa_n\big(\big[\pi'_n(q)\big]_{n+1}\big) \\
&= \kappa_n\big(f\big(\big[\pi'_n(r)\big]_{n+1}\big)\big) \\
&= \kappa_n(f)\big([r]_n\big) \\
&= \kappa_n(f)\big(\kappa_n\big([r]_{n+1}\big)\big) \\
&= \kappa_n(f)\Big(\kappa_n\big(g\big(\big[\pi'_n(p)\big]_{n+1}\big)\big)\Big) \\
&= \kappa_n(f) \circ \kappa_n(g)\big([p]_n\big).
\end{aligned}$$

Now we will show that κ_n distributes over the application function χ. Suppose that $f \in H_{n+1}$ and let $[p]_{n+1}$ be in A_{n+1}. Suppose that $f\big([p]_{n+1}\big) = [q]_{n+1}$. Consider the following:

$$\begin{aligned}
\kappa_n\big(\chi\big(f, [p]_{n+1}\big)\big) &= \kappa_n\big([q]_{n+1}\big) \\
&= [q]_n \\
&= \kappa_n\big(\big[\pi_n'(q)\big]_{n+1}\big) \\
&= \kappa_n \circ f\big(\big[\pi_n'(p)\big]_{n+1}\big) \\
&= \kappa_n(f)\big([p]_n\big) \\
&= \chi\big(\kappa_n(f), \kappa_n\big([p]_{n+1}\big)\big).
\end{aligned}$$

We constructed thus a row

$$\mathfrak{M}_1 \xleftarrow{\kappa_1} \mathfrak{M}_2 \xleftarrow{\kappa_2} \mathfrak{M}_3 \xleftarrow{\kappa_3} \mathfrak{M}_4 \longleftarrow \ldots \tag{11}$$

of models with homomorphisms between them. With this row, we will construct the inverse limit $\mathfrak{M}^\infty$. Let A^∞ and H^∞ be as follows:

$$\begin{aligned}
A^\infty &= \big\{\, (x_n)_n \;\big|\; x_n \in A_n, \kappa_n(x_{n+1}) = x_n \,\big\}, \\
H^\infty &= \big\{\, (f_n)_n \;\big|\; f_n \in H_n, \kappa_n(f_{n+1}) = f_n \,\big\}.
\end{aligned}$$

We will call the elements of these sets projective rows. We will define the operations hereinafter. Let $\star$ be one of $\|, \underline{\|}, |, +$ or $\cdot$. Let $(x_n)_n, (y_n)_n \in A^\infty$. Then we define $\star$ to be

$$(x_n)_n \star (y_n)_n = (x_n \star y_n)_n.$$

It is obvious that $(x_n \star y_n)_n \in A^\infty$. Now we will define the composition of functions. Let $(f_n)_n, (g_n)_n \in H^\infty$. Then we define

$$(f_n)_n \circ (g_n)_n = (f_n \circ g_n)_n.$$

From the fact that κ_n is a homomorphism it follows immediately that

$$(f_n \circ g_n)_n \in H^\infty.$$

Finally, we define the application function. Let $(f_n)_n \in H^\infty$ and let $(x_n)_n$ be in A^∞. Then we define χ to be

$$\chi\big((f_n)_n, (x_n)_n\big) = \big(\chi(f_n, x_n)\big)_n.$$

It will be clear that $\big(\chi(f_n, x_n)\big)_n \in A^\infty$. Now we will define constants of sort H^∞. Firstly we will introduce the encapsulation operator. To prevent any confusion we will label the encapsulation operators in all sets H_n as follows:

$\partial_H^n \in H_n$. We define the encapsulation operator to be $\partial_H = (\partial_H^n)_n$. We will illustrate that $\partial_H \in H^\infty$. For every component of the encapsulation operator, we have of course $\partial_H^n \in H_n$. It remains thus to prove that ∂_H is a projective row. Let thereto $[p]_n$ be in A_n. We easily deduce the following:

$$\begin{aligned}\kappa_n(\partial_H^{n+1})\big([p]_n\big) &= \kappa_n \circ \partial_H^{n+1}\big(\big[\pi'_n(p)\big]_{n+1}\big)\\ &= \big[\partial'_H \circ \pi'_n(p)\big]_n\\ &= \partial_H^n\big([p]_n\big).\end{aligned}$$

So we find that $\kappa_n(\partial_H^{n+1}) = \partial_H^n$. Secondly we will introduce for all $k \geq 1$, the projection operators. We will label the projections in the sets H_n just as above: $\pi_k^n \in H_n$. Now we define the projection operator to be $\pi_k = (\pi_k^n)_n$. We will show that this projection is an element of H^∞. The first condition is satisfied by definition, so let us verify that $(\pi_k^n)_n$ is a projective row. Let $[p]_n$ be in A_n. Then it is easy to see the following:

$$\begin{aligned}\kappa_n(\pi_k^{n+1})\big([p]_n\big) &= \big[\pi'_k \circ \pi'_n(p)\big]_n\\ &= \pi_k^n\big([p]_n\big).\end{aligned}$$

And we find $\kappa_n(\pi_k^{n+1}) = \pi_k^n$. At this point, we will introduce constants in A^∞. Let a be an atomic action. Then we define a row $\big([a]_n\big)_n$. We immediately see, by definition, that this row is in A^∞.

Now let $\mathfrak{M}^\infty$ be the algebra that consists of the sets A^∞ and H^∞, the operators $+, \cdot, \|, \mathbin{\|\!\!\_}, |, \chi$ and $\circ$, and the constants $\big([a]_n\big)_n \in A^\infty$ for all $a \in A$ and the constants $\pi_k, \partial_H \in H^\infty$, for all $k \geq 1$ and $H \subseteq A$. We will define the projections

$$\zeta_n : \mathfrak{M}^\infty \longrightarrow \mathfrak{M}_n$$

as follows. $\zeta_n(x) = x_n$, and $\zeta(f) = f_n$ if $x = (x_n)_n$ and $f = (f_n)_n$. It is very easy to see that for all $n \geq 1$, we have $\kappa_n \circ \zeta_{n+1} = \zeta_n$. It is a well-known fact that such a construction forms the projective or inverse limit. We find thus that $\mathfrak{M}^\infty$ is the projective limit of the row in display (11).

Theorem (3.5.23)

$$\mathfrak{M}^\infty \models ACP_u.$$

See table 3.5 on page 115 for the axioms of ACP_u.

Proof. We know from theorem (3.5.10) that for all $n \geq 1 : \mathfrak{M}_n \models ACP_u$. Forasmuch as the operations on $\mathfrak{M}^\infty$ are defined component by component, we see at once that $\mathfrak{M}^\infty \models ACP_u$. This will end the proof of 3.5.23.

Theorem (3.5.24)

$$\mathfrak{M}^\infty \models RDP, RSP.$$

Proof. We will use the notations that we have already introduced in theorem (3.5.17). So let

$$V = \{x_\alpha : \alpha \in I\}$$

be a set of variables and let

$$E(V) = \left\{\, x_\alpha = t_\alpha\big((x_\beta)_{\beta\in I}\big) : \alpha \in I \,\right\}$$

be a guarded recursive specification. Without loss of generality, we may assume that $E(V)$ is completely guarded; see definition (3.2.15). For all $n \geq 1$ we have constructed in theorem (3.5.17) a solution $([p^n_\alpha])_\alpha$ for $E(V)$ in $\mathfrak{M}_n$. We claim that $\big([p^n_\alpha]_n\big)_n$ is an element of A^∞ for all $\alpha \in I$. This follows immediately from the following:

$$\begin{aligned}
\kappa_n\big(\big[p^{n+1}_\alpha\big]_{n+1}\big) &= \big[p^{n+1}_\alpha\big]_n \\
&= \big[\pi'_n(p^{n+1}_\alpha)\big]_n \\
&= \big[\pi'_n(p^n_\alpha)\big]_n && \text{see (3)} \\
&= [p^n_\alpha]_n.
\end{aligned}$$

We are to verify that for all $\alpha \in I : \big([p^n_\alpha]_n\big)_n = \big(\big[t_\alpha(p^n_\beta)_\beta\big]_n\big)_n$. But we know from the proof of theorem (3.5.17) that for all $n \geq 1$ we have

$$[p^n_\alpha]_n = \big[t_\alpha(p^n_\beta)_\beta\big]_n;$$

so we see that $\mathfrak{M}^\infty \models RDP$, too. Now suppose that for all $\alpha \in I$ we have $\big([q^n_\alpha]_n\big)_n = \big(\big[t_\alpha(q^n_\beta)_\beta\big]_n\big)_n$. Let $n \geq 1$ be fixed. Since RSP is valid in $\mathfrak{M}_n$, we see that $[q^n_\alpha]_n = [p^n_\alpha]_n$, so we also find $\big([p^n_\alpha]_n\big)_n = \big([q^n_\alpha]_n\big)_n$ and $\mathfrak{M}^\infty \models RSP$. This ends the proof of theorem 3.5.24.

Theorem (3.5.25)

$$\mathfrak{M}^\infty \models AIP.$$

Proof. Let $(x_n)_n, (y_n)_n \in A^\infty$. Suppose that for all $k \geq 1$, we have

$$\pi_k\big((x_n)_n\big) = \pi_k\big((y_n)_n\big).$$

Then we find for all $n, k \geq 1 : \pi_k(x_n) = \pi_k(y_n)$. Now fix an $n \geq 1$; we know that $\mathfrak{M}_n \models AIP$, so we find $x_n = y_n$. But this is valid for all $n \geq 1$ and we find thus $(x_n)_n = (y_n)_n$. This is precisely what we wanted to prove.

Theorem (3.5.26)

$$\mathfrak{M}^\infty \models EA.$$

Proof. Since H^∞ is a function space, we immediately have this property. See also the proof of (3.5.20).

Theorem (3.5.27)

$$\mathfrak{M}^\infty \models ODP, OSP.$$

Proof. We will modify the notations of theorem (3.5.21) slightly. In there we defined a sequence of elements

$$\mu_1^r, \ldots, \mu_l^r \in H_n$$

for a fixed $n \geq 1$ and $1 \leq r \leq n$. We will give this sequence of elements an extra label as follows:

$$\mu_1^{n,r}, \ldots, \mu_l^{n,r} \in H_n.$$

We define for all i with $1 \leq i \leq l$ the row μ_i to be $(\mu_i^{n,n})_n$. We claim that

$$\mu_1, \ldots, \mu_l \in H^\infty. \tag{12}$$

To prove (12) we fix an i with $1 \leq i \leq l$. By definition, we see that for all $n \geq 1$ we have $\mu_i^{n,n} \in H_n$, so we have to verify that μ_i is a projective row, that is, we are to show that for all $n \geq 1$

$$\kappa_n(\mu_i^{n+1,n+1}) = \mu_i^{n,n}. \tag{13}$$

First, we will treat the case $n = 1$. Let $[p]_1$ be as in equation (6) of theorem (3.5.21). It is very easy to see that

$$\begin{aligned}\kappa_1(\mu_1^{2,2})\big([p]_1\big) &= \sum_{j=1}^{s}\big[a_j^{(i)}\big]_1 + \sum_{k=1}^{t}\big[b_k^{(i)}\big]_1 \\ &= \mu_1^{1,1}\big([p]_1\big).\end{aligned}$$

Now let henceforward $n > 1$. In order to verify (13) we will need the following intermediate result. For all $1 \leq i \leq l$ and for all $1 \leq r \leq n$, we have

$$\kappa_n(\mu_i^{n+1,r}) = \mu_i^{n,r}. \tag{14}$$

Let $r = 1$ and let $1 \leq i \leq l$ be chosen. Let $[p]_n \in A_n$ be as in equation (6). Consider the following.

$$\begin{aligned}\kappa_n(\mu_i^{n+1,1})\big([p]_n\big) &= \kappa_n \circ \mu_i^{n+1,1}\big(\big[\pi_n'(p)\big]_{n+1}\big) \\ &= \sum_{j=1}^{s}\big[a_j^{(i)}\big]_n + \sum_{k=1}^{t}\big[b_k^{(i)}\big]_n \\ &= \mu_1^{n,1}\big([p]_n\big).\end{aligned}$$

Now let $1 \leq r < n$ and suppose that (14) is proved up to and including r. We will prove it for $r+1$.

$$\begin{aligned}
\kappa_n(\mu_i^{n+1,r+1})\big([p]_n\big) &= \kappa_n \circ \mu_i^{n+1,r+1}\big(\big[\pi'_n(p)\big]_{n+1}\big) \\
&= \sum_{j=1}^{s}\big[a_j^{(i)}\big]_n \cdot \kappa_n(\mu^{n+1,r}_{\sigma(a_j,i)})\big(\big[\pi'_{n-1}(p_j)\big]_n\big) + \sum_{k=1}^{t}\big[b_k^{(i)}\big]_n \\
&= \sum_{j=1}^{s}\big[a_j^{(i)}\big]_n \cdot \mu^{n,r}_{\sigma(a_j,i)}\big(\big[\pi'_{n-1}(p_j)\big]_n\big) + \sum_{k=1}^{t}\big[b_k^{(i)}\big]_n \\
&= \mu_i^{n,r+1}\big(\big[\pi'_n(p)\big]_n\big) \\
&= \mu_i^{n,r+1}\big([p]_n\big).
\end{aligned}$$

This ends the proof of (14). In particular, we find for $r = n$

$$\kappa_n(\mu_i^{n+1,n}) = \mu_i^{n,n}. \tag{15}$$

Now we will verify that equation (13) is valid. Let $[p]_n$ still be as in equation (6). We will calculate the left-hand side of equation (13).

$$\begin{aligned}
\kappa_n(\mu_i^{n+1,n+1})\big([p]_n\big) &= \sum_{j=1}^{s}\big[a_j^{(i)}\big]_n \cdot \kappa_n(\mu^{n+1,n}_{\sigma(a_j,i)})\big(\big[\pi'_{n-1}(p_j)\big]_n\big) + \sum_{k=1}^{t}\big[b_k^{(i)}\big]_n \\
&\overset{(15)}{=} \sum_{j=1}^{s}\big[a_j^{(i)}\big]_n \cdot \mu^{n,n}_{\sigma(a_j,i)}\big(\big[\pi'_{n-1}(p_j)\big]_n\big) + \sum_{k=1}^{t}\big[b_k^{(i)}\big]_n.
\end{aligned}$$

We know that all $\mu_i^{n,n}$ satisfy equations (4) and (5) of theorem (3.5.21). So we find for the right-hand side of equation (13):

$$\begin{aligned}
\mu_i^{n,n}\big([p]_n\big) &= \mu_i^{n,n}\big(\big[\pi'_n(p)\big]_n\big) \\
&= \sum_{j=1}^{s}\big[a_j^{(i)}\big]_n \cdot \mu^{n,n}_{\sigma(a_j,i)}\big(\big[\pi'_{n-1}(p_j)\big]_n\big) + \sum_{k=1}^{t}\big[b_k^{(i)}\big]_n,
\end{aligned}$$

and therewith we deduced that $\mu_i \in H^\infty$. In order to prove the existential part of the theorem, i.e., $\mathfrak{M}^\infty \models ODP$, it suffices to show that for $\mu_1, \ldots, \mu_l$ the following two equations hold.

$$\begin{aligned}
\mu_i\big(([a]_n)_n\big) &= \big(\big[a^{(i)}\big]_n\big)_n, \\
\mu_i\big(([a]_n)_n \cdot (x_n)_n\big) &= \big(\big[a^{(i)}\big]_n\big)_n \cdot \mu_{\sigma(a,i)}\big((x_n)_n\big),
\end{aligned}$$

in which $a \in A$ and $(x_n)_n \in H^\infty$. This is trivial. So we find that $\mathfrak{M}^\infty \models ODP$.

Now we will prove the uniqueness part. Suppose that $\nu_1, \ldots, \nu_l \in H^\infty$ solves the system $E(M)$ in $\mathfrak{M}^\infty$, too. With $\nu_i = (\nu_i^n)_n$. Let $p \in G$. It is easy to infer that for all $n \geq 1$ we have

$$\nu_i^n\big([a]_n\big) = \big[a^{(i)}\big]_n,$$
$$\nu_i^n\big([a]_n \cdot [p]_n\big) = \big[a^{(i)}\big]_n \cdot \nu_{\sigma(a,i)}^n\big([p]_n\big).$$

But because of theorem (3.5.21), we know that $\nu_i^n = \mu_i^{n,n}$. So we find

$$\nu_i = (\nu_i^n)_n = (\mu_i^{n,n})_n = \mu_i.$$

So $\mathfrak{M}^\infty \models$ *OSP*. This will finish the proof of 3.5.27.

Remark (3.5.28)

We will mention briefly some observations about problems with the construction of models for $ACP_{\tau,u}$ and additional principles. The general idea for the construction of these models is that we take for the sort of processes an existing model such as a bisimulation model. For the sort of linear unary operators we take a function space on this model. However, this function space must be chosen with care. If we choose the function space too small we will obtain the situation that *ODP* is not valid. For example, take the empty function space. If there are too many functions in the space we will obtain that *OSP* does not hold. We will give an example due to Jan Bergstra [**11**] of this situation. We are going to show that there are two different functions that satisfy the same linear functional specification. For the sort of processes we take the set of finitely branching rooted labeled trees modulo weak bisimulation $\mathscr{B}$. We will call elements of $\mathscr{B}$ process trees. For the function space we take the bisimulation preserving linear unary functions on $\mathscr{B}$. The function that we will define is called f. We will chose a particular label $l \in A$. The function f renames every label of a process tree into τ but it "temporarily" remembers the original label. Then in every node of the process tree it determines if there is a trace with infinitely many labels that are not equal to τ. If the answer is affirmative the function f will add to these nodes the option to do an l. If all the nodes are treated this way the function will forget the original labels. We will give a typical example: we will show in figure 3.3 how the function f behaves on a^ω.

We will not prove that X1, X2 and X3 are valid for f but we will sketch that it is a solution for the next linear functional specification $E(n)$.

$$E(n) = \{n(a) = \tau : a \in A\} \cup \{n(a \cdot x) = n(a) \cdot n(x) : a \in A\}.$$

First, we see that $f(a) = \tau$. For, in the process tree representing a there are no infinite traces. So the boundary conditions are satisfied. We will only show that $f(a \cdot a^\omega) = f(a) \cdot f(a^\omega)$. In figure 3.3 we see that $f(a^\omega) = l + \tau \cdot f(a^\omega)$.

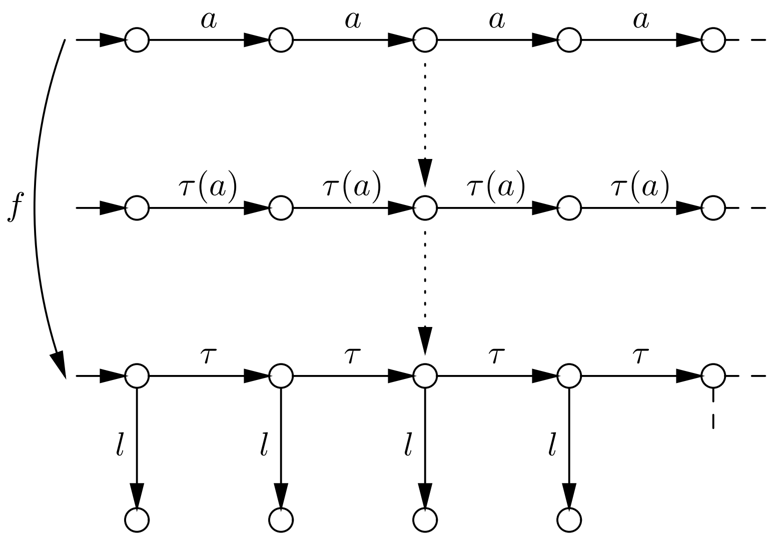

Figure 3.3. A visualization of $a^\omega \mapsto f(a^\omega)$.

Observe that the following derivation uses axiom T2 therefore, the example is specific for weak bisimulation.

$$\begin{aligned} f(a^\omega) &= l + \tau \cdot f(a^\omega) \\ &= l + \tau \cdot f(a^\omega) + \tau \cdot f(a^\omega) \\ &= f(a^\omega) + \tau \cdot f(a^\omega) \\ &= \tau \cdot f(a^\omega) \\ &= f(a) \cdot f(a^\omega). \end{aligned}$$

So the function f will solve the linear functional specification $E(n)$. But on the other hand τ_A will solve this system, too. But clearly we have that $f \neq \tau_A$. This means that the function space that we chosed is too large.

The problem with the function f is that it has an infinite lookahead. At every node it can decide whether or not there was an infinite trace of non-τ actions. If we only look at the functions that have no lookahead we will have a function space in which *ODP* and *OSP* hold. Such functions could be called sequential renamings. The idea is that such a function is specified with the aid of a linear functional specification, which is, in fact, a set of renaming schemes. It starts at the root of a process tree and it renames the labels according to its renaming scheme. The original labels tell the function which renaming scheme must be used for the next node, and so on. These functions are called sequential renamings since after a sequential composition the renaming scheme turns into probably another scheme.

Another observation is that not every model of a process algebra can be extended with an appropriate function space. The following example is also due to Jan Bergstra [**11**]. Consider the subalgebra $\mathscr{A}$ of $\mathscr{B}$ that consists of all the process trees with only a finite number of b-labels. For the function space we take a very large one: the bisimulation preserving linear unary functions on $\mathscr{A}$. The linear functional specification

$$E(n) = \{n(a) = b : a \in A\} \cup \{n(a \cdot x) = n(a) \cdot n(x) : a \in A\}$$

will not have a solution since a solution for the linear functional specification $E(n)$ should map a^ω to b^ω and we clearly have that $b^\omega \notin \mathscr{A}$.

3.6. Applications

In this section we will apply the theory $ACP_{\tau,u}$ on the one hand, by proving a number of basic adversaria in a "new" way. By this we mean an enumeration of straightforward non-related results (at first sight) that are already known. The main difference here is that we will use $ACP_{\tau,u}$ to prove these theorems. In fact, the correlation between these results is the use of auxiliary linear unary operators in order to prove them. On the other hand, it is not only customary to use auxiliary linear unary operators in verifications, but also in specifications. So we will give some examples of specifications with the aid of such auxiliary operators, too, in which we will use *ODP* and *OSP* to introduce the operators needed for the specification. In practice it is a combination of both: specification with the aid of linear unary operators in order to be able to give a verification.

The first example in using an auxiliary operator is a theorem that states: $KFAR_1 \Longrightarrow KFAR_2$. For the definition of *KFAR* we refer to principle (1.2.7). This theorem can be found in [**40**]. In this example, we will use linear unary operators in the body of the proof.

Theorem (3.6.1)

Suppose that $KFAR_1$ holds, then we can derive that $KFAR_2$ holds.

Proof. We will prove this theorem only in the following case. Let u and v be closed terms. Let $i \in I$ and $j \in I$ be internal atomic actions ($I \subseteq A$). Let x and y be processes such that the following holds:

$$x = i \cdot y + u,$$
$$y = j \cdot x + v.$$

We are to show:

$$\tau_I(x) = \tau \cdot \tau_I(u + v).$$

We will use the following abbreviations: $u' = \tau_{\{i\}}(u)$ and $v' = \tau_{\{i\}}(v)$. Consider the guarded recursive specification E_1 below:

$$E_1 = \{\, x_1 = \tau \cdot y_1 + u',\, y_1 = j \cdot x_1 + v' \,\}.$$

(At this point we already see why we consider the closed term case only: the specification E_1 must be without the abstraction operator. Although u' and v' are abbreviations in which the abstraction operator occurs, they are closed terms, so we can eliminate the $\tau_{\{i\}}$ with the axioms concerning the abstraction operator. Hence, we should consider a completely equivalent guarded recursive specification E_1' without the abstraction operator. We will not do so. If we want to prove the general case, we will probably need the notion of guarded recursive specifications with parameters. Then the closed terms u and v can be interpreted as arbitrary processes that are parameters of a guarded recursive specification. The processes u' and v' are in that case just other parameters, hence, we still have a guarded recursive specification without the abstraction operator. The concept of guarded recursive specifications with parameters, can be found in [**32**].)

It is very easy to see that the conditions of theorem (3.4.12) are satisfied, so we see that $\tau_{\{i\}}$ is idempotent. This can be used to see that the unique solution of E_1 is $\big(\tau_{\{i\}}(x), \tau_{\{i\}}(y)\big)$. We will use the fact that $\tau_{\{i\}}$ is idempotent tacitly in the sequel. Now consider the guarded recursive specification E_2:

$$E_2 = \big\{\, x_2 = i \cdot y_2 + u' + v',\ y_2 = j \cdot x_2 + v' \,\big\}.$$

Let (x_2, y_2) be the solution of E_2. Then it is clear that we have the following for $\tau_{\{i\}}(y_2)$ and $\tau_{\{i\}}(x_2)$:

$$\tau_{\{i\}}(y_2) = j \cdot \tau_{\{i\}}(x_2) + v'$$

and

$$\begin{aligned}
\tau_{\{i\}}(x_2) &= \tau \cdot \tau_{\{i\}}(y_2) + u' + v' \\
&= \tau \cdot \tau_{\{i\}}(y_2) + \tau_{\{i\}}(y_2) + u' + v' \\
&= \tau \cdot \tau_{\{i\}}(y_2) + j \cdot \tau_{\{i\}}(x_2) + v' + v' + u' \\
&= \tau \cdot \tau_{\{i\}}(y_2) + j \cdot \tau_{\{i\}}(x_2) + v' + u' \\
&= \tau \cdot \tau_{\{i\}}(y_2) + \tau_{\{i\}}(y_2) + u' \\
&= \tau \cdot \tau_{\{i\}}(y_2) + u'.
\end{aligned}$$

Hence, $\big(\tau_{\{i\}}(x_2), \tau_{\{i\}}(y_2)\big)$ is also a solution for the guarded recursive specification E_1, so by *RSP* we obtain: $\tau_{\{i\}}(x_2) = \tau_{\{i\}}(x)$. With the aid of (3.4.18), it is easy to see that τ_I is a right-absorber for $\tau_{\{i\}}$, that is, we know that $\tau_I \circ \tau_{\{i\}} = \tau_I$. Thus, we achieve:

$$\begin{aligned}
\tau_I(x_2) &= \tau_I \circ \tau_{\{i\}}(x_2) \\
&= \tau_I \circ \tau_{\{i\}}(x) \\
&= \tau_I(x). \qquad (1)
\end{aligned}$$

Now consider the guarded recursive specifications E_3 and E_4:

$$E_3 = \{\, x_3 = i \cdot y_3 + u'' + v'',\ y_3 = \tau \cdot x_3 + u'' + v'' \,\},$$
$$E_4 = \{\, x_4 = i \cdot y_4 + u' + v',\ y_4 = j \cdot x_4 + u' + v' \,\}.$$

We used above the abbreviations $\tau_{\{i,j\}}(u) = u''$ and $\tau_{\{i,j\}}(v) = v''$. With the aid of corollary (3.4.22) it is immediately clear that $\tau_{\{j\}} \circ \tau_{\{i\}} = \tau_{\{i,j\}}$. Now we can derive:

$$\tau_{\{j\}}(x_2) = i \cdot \tau_{\{j\}}(y_2) + u'' + v''$$

and

$$\begin{aligned}
\tau_{\{j\}}(y_2) &= \tau \cdot \tau_{\{j\}}(x_2) + v'' \\
&= \tau \cdot \tau_{\{j\}}(x_2) + \tau_{\{j\}}(x_2) + v'' \\
&= \tau \cdot \tau_{\{j\}}(x_2) + i \cdot \tau_{\{j\}}(y_2) + u'' + v'' + v'' \\
&= \tau \cdot \tau_{\{j\}}(x_2) + i \cdot \tau_{\{j\}}(y_2) + u'' + v'' + u'' + v'' \\
&= \tau \cdot \tau_{\{j\}}(x_2) + \tau_{\{j\}}(x_2) + u'' + v'' \\
&= \tau \cdot \tau_{\{j\}}(x_2) + u'' + v''.
\end{aligned}$$

We see that $\big(\tau_{\{j\}}(x_2), \tau_{\{j\}}(y_2)\big)$ is a solution for E_3. For the solution (x_4, y_4) of E_4 we can deduce easily:

$$\tau_{\{j\}}(x_4) = i \cdot \tau_{\{j\}}(y_4) + u'' + v''$$

and

$$\tau_{\{j\}}(y_4) = \tau \cdot \tau_{\{j\}}(x_4) + u'' + v''.$$

Now we see that $\big(\tau_{\{j\}}(x_4), \tau_{\{j\}}(y_4)\big)$ is also a solution for E_3. Hence, by *RSP* we obtain $\tau_{\{j\}}(x_4) = \tau_{\{j\}}(x_2)$. Completely analogous to the calculation of equation (1), we find:

$$\tau_I(x_4) = \tau_I(x_2). \tag{2}$$

Observe that we use here (3.4.18) since we use that τ_I is a right-absorber for $\tau_{\{j\}}$. Consider the guarded recursive specifications E_5 and E_6:

$$E_5 = \{\, x_5 = j \cdot x_5 + u' + v' \,\},$$
$$E_6 = \{\, x_6 = j \cdot y_6 + u' + v',\ y_6 = j \cdot x_6 + u' + v' \,\}.$$

At this point we want to introduce a linear unary operator. Let n be a function name. Consider the linear functional specification below:

$$\begin{aligned}
E(n) = \{\, n(a) = a \bigm| a \in A \setminus \{i\} \,\} &\cup \{n(i) = j\} \\
&\cup \{\, n(a \cdot x) = n(a) \cdot n(x) \mid a \in A \,\}.
\end{aligned}$$

With the aid of *ODP* we know that there is a valuation φ which solves this specification. Let us say $\varphi(n) = \rho \in F$. With the aid of theorem (3.4.15), we find that $\tau_{\{i\}}$ is a left-absorber for ρ. So for the solution (x_4, y_4) of E_4, we can derive:

$$\begin{aligned} \rho(x_4) &= j \cdot \rho(y_4) + u' + v', \\ \rho(y_4) &= j \cdot \rho(x_4) + u' + v'. \end{aligned}$$

So we see that $\big(\rho(x_4), \rho(y_4)\big)$ is a solution for E_6. On the other hand, if we interchange the order of the equations above, we see that $\big(\rho(y_4), \rho(x_4)\big)$ is a solution for E_6, thus, with the aid of *RSP*, we obtain $\rho(x_4) = \rho(y_4)$. In particular we find:

$$\rho(x_4) = j \cdot \rho(x_4) + u' + v'.$$

Hence, $\rho(x_4)$ is a solution for E_5. Let x_5 be the solution for E_5, then we obtain by *RSP* $\rho(x_4) = x_5$. Use theorem (3.4.18) to see that τ_I is a right-absorber for ρ. We will use this fact in the calculation hereinafter. On the one hand we have:

$$\begin{aligned} \tau_I(x_5) &= \tau_I \circ \rho(x_4) \\ &= \tau_I(x_4) \\ &= \tau_I(x_2) && \text{because of (2)} \\ &= \tau_I(x). && \text{because of (1)} \end{aligned}$$

And on the other hand we may apply $KFAR_1$ to the equation $x_5 = j \cdot x_5 + u' + v'$. This gives the following: (notice that we use the fact that τ_I is a right-absorber for $\tau_{\{i\}}$)

$$\begin{aligned} \tau_I(x_5) &= \tau \cdot \tau_I(u' + v') \\ &= \tau \cdot \big(\tau_I \circ \tau_{\{i\}}(u) + \tau_I \circ \tau_{\{i\}}(v)\big) \\ &= \tau \cdot \big(\tau_I(u) + \tau_I(v)\big) \\ &= \tau \cdot \tau_I(u + v). \end{aligned}$$

Combining these two inferences, we conclude the proof of 3.6.1.

In our next example, we will give a recursive specification of a queue Q over a data set D with input channel 1 and output channel 2. We suppose that the data set D contains more than one datum (see figure 3.4).

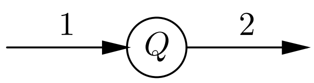

Figure 3.4. A queue Q with input channel 1 and output channel 2.

It is known that an infinite guarded recursive specification of Q can be given by the equations hereinafter:

$$Q = Q_\lambda = \sum_{d \in D} r_1(d) \cdot Q_d, \tag{3}$$

$$Q_{\sigma * d} = s_2(d) \cdot Q_\sigma + \sum_{e \in D} r_1(e) \cdot Q_{e * \sigma * d}, \tag{4}$$

for any word $\sigma \in D^*$ and any $d \in D$. Here, we use $\lambda \in D^*$ for the empty word. The asterisk $(*)$ stands for the concatenation of words. It is known that there is no finite guarded recursive specification over *ACP* which defines the process Q. It is also known that the queue can be specified with the aid of a finite guarded recursive specification, if we allow certain auxiliary linear unary operators (renaming operators). Both results can be found in [**4**]. Below we will state the latter theorem in terms of $ACP_{\tau,u}$.

Theorem (3.6.2)

The queue is definable by a finite guarded recursive specification in $ACP_{\tau,u}$.

Proof. Suppose that we have for the set A of atomic actions

$$\{r_1(d), s_2(d), l(d), u(d) : d \in D\} \subseteq A.$$

Let the communication function be given as follows:

$$\forall\, d \in D : l(d) \mid l(d) = u(d),$$

and all the other communications result in δ. Consider the following linear functional specification for the set of function names $N = \{n, m\}$:

$$\begin{aligned} E(N) = {} & \big\{\, n\big(u(d)\big) = s_2(d) : d \in D \,\big\} \cup \big\{\, n\big(l(d)\big) = \delta : d \in D \,\big\} \\ & \cup \big\{\, n(a) = a : a \in A \setminus \big(u(D) \cup l(D)\big) \,\big\} \\ & \cup \big\{\, m\big(s_2(d)\big) = l(d) : d \in D \,\big\} \cup \big\{\, m(a) = a : a \in A \setminus s_2(D) \,\big\} \\ & \cup \big\{\, f(a \cdot x) = f(a) \cdot f(x) \;\big|\; f \in N, a \in A \big\}. \end{aligned}$$

We used above the following abbreviations: $u(D) = \{u(d) : d \in D\}$, $l(D) = \{l(d) : d \in D\}$ and $s_2(D) = \{s_2(d) : d \in D\}$. According to *ODP*, there is a valuation $\varphi : N \longrightarrow F$. Let us say $\varphi(n) = \nu \in F$ and $\varphi(m) = \mu \in F$. Consider the following finite guarded recursive specification:

$$\begin{aligned} R &= \sum_{d \in D} r_1(d) \cdot \nu\big(\mu(R) \parallel s_2(d) \cdot Z\big), \\ Z &= \sum_{d \in D} l(d) \cdot Z. \end{aligned}$$

Now let $\sigma \in D^*$, then we define the process R_σ inductively as displayed below:

$$\begin{aligned} R_\lambda &= R, \\ R_{\sigma * d} &= \nu\big(\mu(R_\sigma) \parallel s_2(d) \cdot Z\big). \end{aligned}$$

At this point, we will prove the following claim:

$$R_\sigma = \nu\big(\mu(R_\sigma) \parallel Z\big). \tag{5}$$

We will verify this by showing that both left and right-hand side of equation (5) are solutions for the same guarded recursive specification and then we will apply *RSP*. In order to prove this, we need another intermediate result:

$$\forall\, \sigma \in D^*, \forall\, d \in D: \quad \nu\big(\mu(R_\sigma) \mathbin{⫫} s_2(d) \cdot Z\big) = \sum_{e \in D} r_1(e) \cdot R_{e * \sigma * d}. \tag{6}$$

We will prove (6) with induction on the length of the word σ. First let $d \in D$ be arbitrarily chosen and let $\sigma = \lambda$. Consider the following calculation:

$$\begin{aligned} \nu\big(\mu(R_\lambda) \mathbin{⫫} s_2(d) \cdot Z\big) &= \nu\Big(\sum_{e \in D} r_1(e) \cdot \mu(R_e) \mathbin{⫫} s_2(d) \cdot Z\Big) \\ &= \sum_{e \in D} r_1(e) \cdot \nu\big(\mu(R_{e*\lambda}) \parallel s_2(d) \cdot Z\big) \\ &= \sum_{e \in D} r_1(e) \cdot R_{e*\lambda*d}. \end{aligned}$$

Hence, for $\sigma = \lambda$ equation (6) is proved. Now let $d \in D$ and let $\sigma = \rho * f \in D^*$ for some $f \in D$, such that equation (6) is proved for $\rho \in D^*$. We show that (6) holds for σ:

$$\begin{aligned} \nu\big(\mu(R_\sigma) \mathbin{⫫} s_2(d) \cdot Z\big) &= \nu\Big(\mu \circ \nu\big(\mu(R_\rho) \parallel s_2(f) \cdot Z\big) \mathbin{⫫} s_2(d) \cdot Z\Big) \\ &= \nu\Big(\mu \circ \nu\big(\mu(R_\rho) \mathbin{⫫} s_2(f) \cdot Z\big) \mathbin{⫫} s_2(d) \cdot Z\Big) \\ &\qquad + \nu\Big(\mu \circ \nu\big(s_2(f) \cdot Z \mathbin{⫫} \mu(R_\rho)\big) \mathbin{⫫} s_2(d) \cdot Z\Big) \\ &= \nu\Big(\mu \circ \nu\big(\mu(R_\rho) \mathbin{⫫} s_2(f) \cdot Z\big) \mathbin{⫫} s_2(d) \cdot Z\Big) + \delta \\ &= \nu\Big(\mu\Big(\sum_{e \in D} r_1(e) \cdot R_{e*\rho*f}\Big) \mathbin{⫫} s_2(d) \cdot Z\Big) \\ &= \sum_{e \in D} r_1(e) \cdot \nu\big(\mu(R_{e*\sigma}) \parallel s_2(d) \cdot Z\big) \\ &= \sum_{e \in D} r_1(e) \cdot R_{e*\sigma*d}. \end{aligned}$$

This ends the proof of our intermediate result stated in (6). Let us now calculate the left-hand side of equation (5), first we take $\sigma = \lambda$:

$$\begin{aligned} R_\lambda &= R \\ &= \sum_{d\in D} r_1(d) \cdot \nu\big(\mu(R_\lambda) \parallel s_2(d) \cdot Z\big) \\ &= \sum_{d\in D} r_1(d) \cdot \nu\big(\mu(R) \parallel s_2(d) \cdot Z\big) \\ &= \sum_{d\in D} r_1(d) \cdot R_d. \end{aligned}$$

Now we will drop the assumption of σ being the empty word.

$$\begin{aligned} R_{\sigma*d} &= \nu\big(\mu(R_\sigma) \parallel s_2(d) \cdot Z\big) \\ &= \nu\big(\mu(R_\sigma) \mathbin{⌊\!⌊} s_2(d) \cdot Z\big) + s_2(d) \cdot \nu\big(\mu(R_\sigma) \parallel Z\big) \\ &= \sum_{e\in D} r_1(e) \cdot R_{e*\sigma*d} + s_2(d) \cdot \nu\big(\mu(R_\sigma) \parallel Z\big). \end{aligned} \tag{7}$$

Consider the guarded recursive specification G below:

$$G \;=\; \Big\{ X_\lambda = \sum_{d\in D} r_1(d) \cdot X_d, \quad X_{\sigma*d} = \sum_{e\in D} r_1(e) \cdot X_{e*\sigma*d} + \nu\big(\mu(X_\sigma) \parallel Z\big) \;\Big|\; d \in D, \sigma \in D^* \Big\}.$$

Then it is obvious that, if we put $X_\sigma = R_\sigma$ for all $\sigma \in D^*$, this system is a solution for G. Observe that the linear unary operators, appearing in the guarded recursive specification G, are concrete operators; confer remark (3.2.19). Now we will prove that the right-hand side of equation (5) is also a solution for the specification G. Therefore, we will introduce one more abbreviation:

$$S_\sigma = \nu\big(\mu(R_\sigma) \parallel Z\big).$$

Consider the calculation below for S_λ:

$$\begin{aligned} S_\lambda &= \nu\big(\mu(R_\lambda) \mathbin{⌊\!⌊} Z\big) \\ &= \nu\Big(\mu\Big(\sum_{d\in D} r_1(d) \cdot \nu\big(\mu(R_\lambda) \parallel s_2(d) \cdot Z\big)\Big) \mathbin{⌊\!⌊} Z\Big) \\ &= \nu\Big(\sum_{d\in D} r_1(d) \cdot \mu \circ \nu\big(\mu(R_\lambda) \parallel s_2(d) \cdot Z\big) \mathbin{⌊\!⌊} Z\Big) \\ &= \sum_{d\in D} r_1(d) \cdot \nu\big(\mu \circ \nu\big(\mu(R_\lambda) \parallel s_2(d) \cdot Z\big) \parallel Z\big) \end{aligned}$$

$$
\begin{aligned}
&= \sum_{d \in D} r_1(d) \cdot \nu\big(\mu(R_d) \parallel Z\big) \\
&= \sum_{d \in D} r_1(d) \cdot S_d.
\end{aligned}
$$

Now we calculate $S_{\sigma * d}$:

$$
\begin{aligned}
S_{\sigma * d} &= \nu\big(\mu(R_{\sigma * d}) \parallel Z\big) \\
&= \sum_{e \in D} r_1(e) \cdot \nu\big(\mu(R_{e * \sigma * d}) \parallel Z\big) + s_2(d) \cdot \nu\big(\mu(S_\sigma) \parallel Z\big) \qquad \text{use (7)} \\
&= \sum_{e \in D} r_1(e) \cdot S_{e * \sigma * d} + s_2(d) \cdot \nu\big(\mu(S_\sigma) \parallel Z\big)
\end{aligned}
$$

It is clear that, if we put for all $\sigma \in D^* : X_\sigma = S_\sigma$, this system is also a solution for G. Hence, by *RSP*, we conclude the proof of equation (5). Now we find for the defining equations for R_σ:

$$
\begin{aligned}
R_\lambda &= \sum_{d \in D} r_1(d) \cdot R_d, \\
R_{\sigma * d} &= \sum_{e \in D} r_1(d) \cdot R_{e * \sigma * d} + s_2(d) \cdot R_\sigma.
\end{aligned}
$$

But these are the defining equations of the process $Q = Q_\lambda$; see equations (6) and (7). Thus, again using *RSP*, we obtain $Q = R_\lambda$. This ends the proof of 3.6.2.

At this point, we will give an example in which we will use the operator definition principle in order to specify a communication network. In [**4**] this particular example is specified with an auxiliary operator, which is called the localization operator. In fact, it is the composition of a renaming operator and an abstraction operator. We will handle this in a slightly different way. We will combine three operators in one linear unary operator instead of two: a renaming operator (which is the auxiliary part), an abstraction operator and an encapsulation operator. In this network, we will have an environment E, a sender S and a receiver R. We will have four channels 1–4. We will depict this network in figure 3.5.

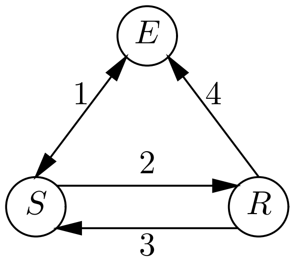

Figure 3.5. A communication network.

Let D be a finite set of data and let $ack \notin D$ be an acknowledgement. We will give the recursive equations for the sender S and the receiver R.

$$\begin{aligned} S &= \sum_{d \in D} r_1(d) \cdot s_2(d) \cdot r_3(ack) \cdot s_1(ack) \cdot S, \\ R &= \sum_{d \in D} r_2(d) \cdot s_4(d) \cdot s_3(ack) \cdot R. \end{aligned}$$

The sender and the receiver both communicate with the environment via channels 1 and 4. The environment E can send data along 1, receive an ack along 1, or receive data along 4. Thus, we can put for the recursive specification for E:

$$E = \Big(\sum_{d \in D} s_1(d) + \sum_{d \in D} r_4(d) + r_1(ack) \Big) \cdot E.$$

The behaviour of the environment is that we first must have an $s_1(d)$, then an $r_4(d)$ and then an $r_1(ack)$ before the next $s_1(e)$ can follow. This ordering of atomic actions is not expressed in the definition of E. We will use an auxiliary linear unary operator to express this behaviour. We assume for the set of atomic actions A the following:

$$\begin{aligned} A = \big\{\, & r_i(d), s_i(d), c_i(d) : d \in D,\ i = 1, 2, 4 \,\big\} \\ & \cup \big\{\, r_i(ack), s_i(ack), c_i(ack) : i = 1, 3 \,\big\}. \end{aligned}$$

Next we will give, for a function name n, the following linear functional specification:

$$\begin{aligned} E(n) = \big\{\, & n\big(r_i(x)\big) = n\big(s_i(x)\big) = \delta \bigm| i = 1, 2, 4,\ x \in D \cup \{ack\} \big\} \\ & \cup \big\{\, n\big(c_1(x)\big) = s_1(x) \bigm| x \in D \cup \{ack\} \big\} \\ & \cup \big\{\, n\big(c_2(d)\big) = \tau \bigm| d \in D \,\big\} \\ & \cup \big\{\, n\big(c_4(d)\big) = r_4(d) \bigm| d \in D \,\big\} \\ & \cup \big\{ n\big(c_3(ack)\big) = \tau \big\} \\ & \cup \big\{\, n(a \cdot x) = n(a) \cdot n(x) \bigm| a \in A \,\big\}. \end{aligned}$$

With the aid of *ODP*, we know that there is a valuation φ that solves this system. Let us say $\varphi(n) = \nu \in F$ (notice that this linear unary operator is an example of an abstracting operator; see definition (3.2.12) on page 76). We can now state the following theorem in which we will use the notations that we have introduced above. This theorem says that we will obtain the actions of E in the right order.

Theorem (3.6.3)

$$\nu(E \parallel S \parallel R) = \sum_{d\in D} s_1(d)\cdot r_4(d)\cdot r_1(ack)\cdot \nu(E \parallel S \parallel R).$$

Proof. This consists of the following straightforward calculation:

$$\begin{aligned}
\nu(E \parallel S \parallel R) &= \nu\big((E \mid S) \mathbin{\lfloor\!\lfloor} R\big)\\
&= \sum_{d\in D} s_1(d)\cdot \nu\big(s_2(d)\cdot r_3(ack)\cdot s_1(ack)\cdot S \parallel E \parallel R\big)\\
&= \sum_{d\in D} s_1(d)\cdot \nu\big(\big(s_2(d)\cdot r_3(ack)\cdot s_1(ack)\cdot S \mid R\big) \mathbin{\lfloor\!\lfloor} E\big)\\
&= \sum_{d\in D} s_1(d)\cdot \tau\cdot \nu\big(s_4(d)\cdot s_3(ack)\cdot R \parallel r_3(ack)\cdot s_1(ack)\cdot S \parallel E\big)\\
&= \sum_{d\in D} s_1(d)\cdot \nu\big(\big(s_4(d)\cdot s_3(ack)\cdot R \mid E\big) \mathbin{\lfloor\!\lfloor} r_3(ack)\cdot s_1(ack)\cdot S\big)\\
&= \sum_{d\in D} s_1(d)\cdot r_4(d)\cdot \nu\big(s_3(ack)\cdot R \parallel E \parallel r_3(ack)\cdot s_1(ack)\cdot S\big)\\
&= \sum_{d\in D} s_1(d)\cdot r_4(d)\cdot \nu\big(\big(s_3(ack)\cdot R \mid r_3(ack)\cdot s_1(ack)\cdot S\big) \mathbin{\lfloor\!\lfloor} E\big)\\
&= \sum_{d\in D} s_1(d)\cdot r_4(d)\cdot \tau\cdot \nu\big(R \parallel s_1(ack)\cdot S \parallel E\big)\\
&= \sum_{d\in D} s_1(d)\cdot r_4(d)\cdot \nu\big(\big(E \mid s_1(ack)\cdot S\big) \mathbin{\lfloor\!\lfloor} R\big)\\
&= \sum_{d\in D} s_1(d)\cdot r_4(d)\cdot s_1(ack)\cdot \nu(E \parallel S \parallel R).
\end{aligned}$$

In the next example, we will describe a process P, which can be in different states: we will specify a process with parallel input. More on this subject can be found in chapter 2. Suppose we have a finite data set D consisting of two elements 0 and 1. We have two input channels: 1 and 2, and one output channel: 3. The process P reads in an arbitrary order the data of the input channels 1 and 2 and sends along channel 3 the sum of these data modulo 2 (and then it starts all over again). We can think of P as a simulation of a(n) `xor`-port with the aid of process algebra. We will depict this process in figure 3.6.

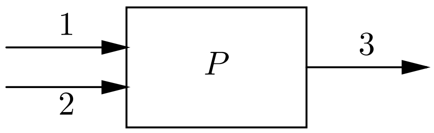

Figure 3.6. Simulation of a(n) `xor`-port.

We will first describe the process P without the merge. We assume that we have the following set of atomic actions:

$$\begin{aligned} A &= \{r_i'(j) : i = 1,2 \ j = 0,1\} \cup \{t\} \\ &\qquad \cup \{r_i(j) : i = 1,2 \ j = 0,1\} \cup \{s_3(j) : j = 0,1\} \\ &= \{r_i'(j) : i = 1,2 \ j = 0,1\} \cup \{t\} \cup B, \end{aligned}$$

with $B = \{r_i(j) : i = 1,2 \ j = 0,1\} \cup \{s_3(j) : j = 0,1\}$. Consider the recursive specification of the process P below:

$$\begin{aligned} P = &\sum_{i=0,1} r_1(i) \cdot \sum_{j=0,1} r_2(j) \cdot s_3(i + j \bmod 2) \cdot P \\ &+ \sum_{l=0,1} r_2(l) \cdot \sum_{k=0,1} r_1(k) \cdot s_3(k + l \bmod 2) \cdot P. \end{aligned}$$

The problem with this specification is that we actually want to denote this process with a merge. For it has parallel input. But if we do so, we will obtain difficulties with the scope of the sum signs. We could write the specification above as follows:

$$P = \Big(\sum_{i=0,1} r_1(i) \parallel \sum_{j=0,1} r_2(j) \Big) \cdot s_3(i + j \bmod 2) \cdot P.$$

Since the variable i under the first sum sign is bounded, we can alter it without changing the meaning of the sum. But the i that occurs in the send action will remain the same. So the meaning of the entire formula will change. We will use an auxiliary unary operator to solve this problem. Let

$$N = \{n_{k,l} : k, l = 0,1\}$$

be a set of function names. Let $E(N)$ be the following linear functional specification.

$$\begin{aligned} E(N) = &\left\{ n_{k,l}\big(r_i'(j)\big) = r_i(j) : j,k,l = 0,1 \ i = 1,2 \right\} \\ &\cup \left\{ n_{k,l}(t) = s_3(k + l \bmod 2) : k,l = 0,1 \right\} \\ &\cup \{n(a) = a : a \in B\} \\ &\cup \left\{ n_{k,l}\big(r_1'(j) \cdot x\big) = r_1(j) \cdot n_{j,l}(x) : j,k,l = 0,1 \right\} \\ &\cup \left\{ n_{k,l}\big(r_2'(j) \cdot x\big) = r_2(j) \cdot n_{k,j}(x) : j,k,l = 0,1 \right\} \\ &\cup \left\{ n_{k,l}(t \cdot x) = s_3(k + l \bmod 2) \cdot n_{0,0}(x) : k,l = 0,1 \right\} \\ &\cup \{n_{k,l}(a \cdot x) = a \cdot n_{k,l}(x) : a \in B \ k,l = 0,1\}. \end{aligned}$$

With the aid of *ODP*, we know that there is a valuation $\varphi : N \longrightarrow F$ which solves this system. Let us say $\varphi(n_{k,l}) = \nu_{k,l} \in F$ with $k,l \in \{0,1\}$. Subsequently, we will give a specification of a process X.

$$X = \Big(\sum_{i=0,1} r_1'(i) \parallel \sum_{j=0,1} r_2'(j) \Big) \cdot t \cdot X.$$

With the notations that we have introduced above we can now state the following theorem.

Theorem (3.6.4)

$$\nu_{0,0}(X) = P.$$

Proof. Consider the following calculation.

$$\begin{aligned}
\nu_{0,0}(X) &= \nu_{0,0}\Big(\sum_{i=0,1} r_1'(i) \cdot \sum_{j=0,1} r_2'(j) \cdot t \cdot X\Big) \\
&\qquad + \nu_{0,0}\Big(\sum_{l=0,1} r_2'(l) \cdot \sum_{k=0,1} r_1'(k) \cdot t \cdot X\Big) \\
&= \sum_{i=0,1} r_1(i) \cdot \nu_{i,0}\Big(\sum_{j=0,1} r_2'(j) \cdot t \cdot X\Big) \\
&\qquad + \sum_{l=0,1} r_2(l) \cdot \nu_{0,l}\Big(\sum_{k=0,1} r_1'(k) \cdot t \cdot X\Big) \\
&= \sum_{i=0,1} r_1(i) \cdot \sum_{j=0,1} \cdot \nu_{i,j}(t \cdot X) \\
&\qquad + \sum_{l=0,1} r_2(l) \cdot \sum_{k=0,1} r_1(k) \cdot \nu_{k,l}(t \cdot X) \\
&= \sum_{i=0,1} r_1(i) \cdot \sum_{j=0,1} \cdot s_3(i + j \bmod 2) \cdot \nu_{0,0}(X) \\
&\qquad + \sum_{l=0,1} r_2(l) \cdot \sum_{k=0,1} r_1(k) \cdot s_3(k + l \bmod 2) \cdot \nu_{0,0}(X).
\end{aligned}$$

With the aid of *RSP* we see that $\nu_{0,0}(X) = P$. This will end the verification of theorem 3.6.4.

3.7. Generalizations

In this section we will mention briefly some generalizations of $ACP_{\tau,u}$. In particular, we will point out that the definition of a linear functional specification can be generalized easily such that, with the aid of *ODP* and *OSP*, we can introduce more linear unary operators. The first generalization that can be thought of is an "exit" possibility. That is, we allow functional equations of the following type:

$$f(a \cdot x) = f(a).$$

Then we are able to specify (operators that behave like) the projection operators, with the aid of a linear functional specification. The second generalization that we can think of is that we have more "liberal" boundary conditions: we would sometimes send an atomic action to a closed term. To exemplify this, we will give hereinafter a definition of a linear functional specification that accommodates both generalizations.

Definition (3.7.1)

Let N be a set of function names. A linear functional specification $E(N)$ for N is a set of the following form:

$$E(N) = \{r_{n,a} : n \in N, a \in A\} \cup \{e_{n,a} : n \in N, a \in A\}. \tag{1}$$

Both $r_{n,a}$ and $e_{n,a}$ are equations. Now fix an $a \in A$ and an $n \in N$, then we will define the two equations $r_{n,a}$ and $e_{n,a}$ for the pair (n,a). The first equation $r_{n,a}$ is called a boundary condition and has the following form: there are closed terms $\{t_k : k \in K\}$ and $\{s_l : l \in L\}$ without linear unary operators, for certain finite disjoint sets K and L, such that

$$r_{n,a} \equiv \chi(n,a) = \sum_{k \in K} t_k + \sum_{l \in L} s_l. \tag{2}$$

The equation $e_{n,a}$ is called a linear functional equation and it is of the following form:

$$e_{n,a} \equiv \chi(n, a \cdot x) = \sum_{k \in K} t_k \cdot \chi(n_k, x) + \sum_{l \in L} s_l, \tag{3}$$

in which $\{n_k : k \in K\} \subseteq N$. If $K = \emptyset$, we omit the first sum sign in both equations (2) and (3) and if $L = \emptyset$, we leave out the second sum sign in (2) and (3).

It will be clear that this definition is a generalization of definition (3.2.2), for let $L = \emptyset$ (this disables the "exit" possibility), let $|K| = 1$ and let the closed term $t_k \in A \cup \{\delta, \tau\}$.

Since, for the generalized state operator*, we have

$$\Lambda_s^m(a) = \sum_{b \in a(m,s)} b,$$

we see that we can specify this operator with a linear functional specification of definition (3.7.1). We will not need the "exit" possibility here.

It is also possible to consider operators $f : P \longrightarrow Q$. Where we have a binary operator "+" on the elements of Q. If we take, for example, $Q = \mathscr{P}(A)$ we can define the binary operator $+$ to be the union of sets. We must change the definition of a linear functional specification in a radical way. For we might want to have functional equations that look like

$$f(a \cdot x) = f(a) + f(x).$$

* see definition (1.3.4)

Examples of such operators are: the trace operator and the alphabet of a process, which can be found in [**4**]. An example of an auxiliary operator of this "type" can be found in chapter 2. It is the collection of all the registers that occur in a process x and it is abbreviated: $\mathrm{reg}(x)$. We will not discuss this type of generalization any further.

Concluding, we can say that the notion of a linear functional specification can be generalized easily, such that we can handle more and more linear unary operators.

3.8. Conclusions

In this chapter we have presented a way to introduce linear unary operators. Moreover, $ACP_{\tau,u}$ is a first attempt to unify many other axiom systems. In section 3.6, all the examples were originally specified in different axiom systems. While in this chapter, these examples are studied in $ACP_{\tau,u}$.

The first example concerning $KFAR_n$, was specified in ACP_τ with renaming operators. This example showed us that the principles *ODP* and *OSP* were not only used to specify an auxiliary linear unary operator in order to give a verification, but we also employed it indirectly: we intensively referred to some (absorption) theorems that were proved, using the principles *ODP* and *OSP*. Another matter that we discovered in studying this example, was that the notion of a guarded recursive specification had to be adapted; see for details remark (3.2.19).

The second example was specified in *ACP* with renaming operators. These renaming operators were more "elementary" than the renaming operators that were defined in [**40**]. A remarkable fact is that in this example we find linear unary operators inside a guarded recursive specification. As they are so-called concrete operators, they are of no harm in the guarded recursive specification.

The next example uses *ACP* with the localization operator. It is an operator that consists of two parts: there is an encapsulation part and there is a "renaming" part. The localization operator is in fact the composition of a renaming operator and an encapsulation operator. We treated this operator in a slightly different way: we made a linear unary operator with three components; besides the two mentioned hereinbefore, we adapted an abstraction part, too. This is done to create an example of an abstracting operator and to make things a little more compact.

The last example was originally specified in ACP_τ and the register operator (see chapter 2). To shorten the proof of theorem (3.6.4), we simplified the situation somewhat: we "reset" the `xor`-port each time after it sent the output along channel 3.

Theorem (3.4.29) was originally formulated with the aid of *ACP* with the (simple) state operator, as can be seen in the remark subsequent to this theorem. We formulate and prove it, in $ACP_{\tau,u}$.

We find thus that $ACP_{\tau,u}$ with of course *ODP* and *OSP*, is a theory that handles all these different cases. So it can be seen as a theory that unifies the other theories. We are not ready, however, since there are examples of linear unary operators that we cannot describe with this theory yet. We can mention the generalized state operator here. The main reason for this "lack", is the definition of a linear functional specification. We kept this definition as simple as possible for heuristic reasons.

A matter that we did not discuss here is the research on non-linear unary operators, such as the priority operator; see section 1.4. It is the author's opinion that it is worthwhile doing some further investigations on both linear and non-linear unary operators.

Chapter 4

On Induction Principles

WE will discuss a theory that is a generalization of the theory of chapter 3, where we developed the theory in its most elementary form. We will explore one of the suggested generalizations in section 3.7, which is the so called "exit" possibility. This means that we are able to reason on projection operators and linear unary operators that behave like projection operators. In fact, we will consider a more general type of projection operators than the usual projection operators. With the generalized projection operators we will propose an induction principle that turns out to be useful in protocol verifications.

The theory in chapter 3 has been developed to obtain a uniform approach for the introduction of linear unary operators in process algebras. Instances of such operators are frequently found in, for example, communication protocols. In many cases a new verification demands a new set of operators. So for each new verification a new axiom system must be given. Moreover, a number of lemmas concerning the brand new operators have to be proved, or, if that fails, additional axioms on these operators have to be added, for instance, the conditional alphabet axioms; see section 1.5. Although these operators are all different, they share certain common properties, which calls for a unifying framework in which it will be possible to introduce these operators and to reason about them in order to save time and effort.

We will prove some general theorems on linear unary operators. Moreover, we will apply the theory by proving that a fair FIFO queue satisfies a criterion for protocol correctness where we will use the induction principle. This is not an individual case: we will show that a wide range of communication protocols in which unbounded queues appear can be verified using the same methods.

4.1. Introduction

We will consider the algebra of communicating processes with abstraction and linear unary operators. The abbreviation for this axiom system will be $ACP_{\tau,u}$. The τ stands for Milner's silent action [**36**] and the u for unary (in linear unary operator). The theory $ACP_{\tau,u}$ is an extension of the algebra of communicating processes with abstraction ACP_τ that has been studied in [**12**]; see for a short introduction chapter 1. In chapter 3 the idea of a two sorted algebra is proposed and it is shown that this is a proper way to obtain a uniform approach for the introduction of linear unary operators and moreover, a number of examples illustrate that $ACP_{\tau,u}$ is a unifying framework for many other theories that consist of ACP or ACP_τ with some additional linear unary operators.

In this chapter we will study the generalization of the theory in which it is possible to specify the projection operators, as well. The projection operators can be found for the first time in process algebra in [**14**]. Due to the systematic research on linear unary operators more general projection operators are proposed instead of the "classical" or *full* projection operators. The approximation induction principle (*AIP*) accompanies the full projection operators; see section 1.2. In this chapter we propose its equivalent for the generalized projection operators. In fact, we propose a number of induction principles: one for each subset of the finite set of atomic actions. If we take this subset to be the set of atomic actions itself we will regain the old induction principle *AIP*. We will show that these principles can be very useful in verifications of correctness theorems on communication protocols.

We will briefly mention the overall structure of this chapter. In section 4.2 we will repeat the main points of the axiomatic framework $ACP_{\tau,u}$ as it is already described in chapter 3 albeit that certain matters are generalizations of the corresponding ones in chapter 3. In the subsequent section 4.3 we will prove some general theorems on linear unary operators. The items in this section are both a continuation and a generalization of theorems that have been studied in section 3.4. In this section we will focus on theorems in combination with the projection operators; for general theorems without projection operators we refer to section 3.4. An important theorem in section 4.3 is the result that the abstraction operator and the projection operators commute under certain circumstances. The proof of this theorem is trivial but its consequences are not. In section 4.4 we will see this when we will explain how this theory can be used in protocol verifications. We will show that a fair FIFO queue with an unbounded capacity satisfies an algebraic criterion for protocol correctness; this is an old problem in process algebra and it has been first studied in [**13**]. With the methods that we use in this proof we will verify three alternating bit protocols with a time out that can take place at any moment. In section 4.5 we will state some conclusions and recommendations for the further development of this theory and we mention a number of protocols that can be verified in the same way as the ones in section 4.4.

4.2. Definitions

In this section we will give the signature and the axioms of $ACP_{\tau,u}$ as it will be used in this chapter. We will mention five principles (with accompanying definitions): the Operator Definition Principle *ODP*, the Operator Specification Principle *OSP*, the Recursive Definition Principle *RDP*, the Recursive Specification Principle *RSP* and Koomen's Fair Abstraction Rule $KFAR_n$. The first two *ODP* and *OSP* can be found in chapter 3. The principles *RDP* and *RSP* can be found in chapter 1. Finally, $KFAR_n$ can also be found in chapter 1.

We will also mention a sixth principle: the Generalized Induction Principle *GIP*. The principle *GIP* is new and is the subject of research in this chapter. It is a generalization of the Approximation Induction Principle AIP^- that can be found in [**21**]; see also [**15**]. *RDP*, *RSP* and *KFAR* were introduced in papers about protocol verification. *AIP* is not used in protocol verifications. The reason for that is that the operators occurring in *AIP* do not commute with an important linear unary operator: the abstraction operator. Using *GIP* will by-pass this problem. We will show this in the next section (see theorem (4.3.12)).

First we will provide a picture of the signature of $ACP_{\tau,u}$. The difference between figure 4.1 and figure 3.1 on page 71 is the set of constants of sort F. It is extended. In figure 3.1 we have besides the encapsulation operator and the abstraction operator all the full projection operators. In figure 4.1 we have also the identity map id and we have for each subset $J \subseteq A$ a set of projection operators.

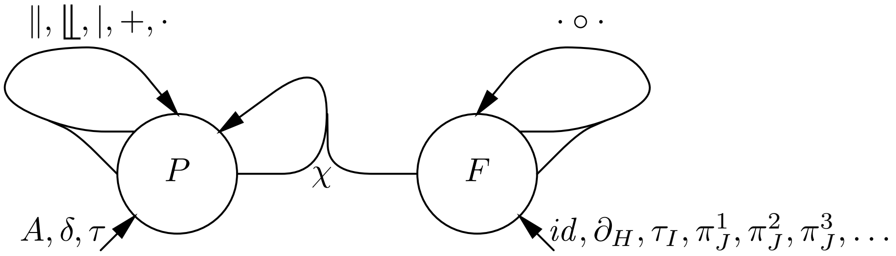

Figure 4.1. Graphical representation of the signature.

Now we will explain this figure. As we can see it is a two-sorted algebra. The first sort that we consider is the sort P of processes. We have a set of constants or atomic actions A in P. There are two so-called special constants in P: deadlock, which is denoted by δ and the silent step, which is abbreviated by τ. We have five binary operators with both arguments in P (they are all infix operators):

merge: $\quad \| : P \times P \longrightarrow P,$

left-merge:	$\mathbin{\lfloor\!\lfloor} : P \times P \longrightarrow P,$
communication-merge:	$\mid : P \times P \longrightarrow P,$
sequential composition:	$\cdot : P \times P \longrightarrow P,$
alternative composition:	$+ : P \times P \longrightarrow P.$

Now we will consider the second one: the sort F of functions or linear unary operators. First, we will discuss the constants of this sort. We have a constant id, which stands for the identity. For all $H \subseteq A$ we have a constant $\partial_H \in F$. We will refer to this constant as the encapsulation operator. For all $I \subseteq A$ we have a constant $\tau_I \in F$, which is called the abstraction operator. Furthermore, we have for all $n \geq 1$ and for all $J \subseteq A$ a constant $\pi_J^n \in F$. This constant will be called the (nth) projection operator. We have only one binary operator with both arguments of sort F:

$$\circ : F \times F \longrightarrow F.$$

This operator will be called the composition of functions. Finally, we have a binary operator that "connects" the two sorts P and F. It is called the apply function:

$$\chi : F \times P \longrightarrow P.$$

This will end the discussion on the signature. Afterwards, we will give the axiom system of $ACP_{\tau,u}$ as it is stated in chapter 3 albeit that we will have other axioms concerning the projection operators. We will use the following notational conventions for the symbols that appear in table 4.1 (page 155): a, b and c are atomic actions, or δ; x, y, z are processes; γ is a special constant, that is, deadlock or silent step; $n \geq 1$; H, I and J are subsets of A; and finally, f, g and h are linear unary operators.

We will formulate an extensionality axiom, which states that linear unary operators that coincide on all processes are indeed the same. We will not use this axiom since we will introduce the notion of a linear functional specification and some principles that allow us to reason about linear unary operators in a comfortable way, but there is no reason to exclude extensionality, so here it is anyway.

Axiom (4.2.1) *Extensionality*

Let f and g be linear unary operators. If for all processes $x \in P : \chi(f, x) = \chi(g, x)$, then $f = g$. We will use the abbreviation EA for this axiom.

Hereinafter, we will define the notion of a linear functional specification. This concept is already defined in chapter 3. We will use a generalized form in which it will be possible to specify the "generalized" projection operators $\pi_J^n \in F$. This form is suggested in section 3.7.

A1	$x + y = y + x$	$\chi(\tau_I, a) = a, \text{ if } a \notin I$	TI1
A2	$x + (y + z) = (x + y) + z$	$\chi(\tau_I, a) = \tau, \text{ if } a \in I$	TI2
A3	$x + x = x$	$\chi(\tau_I, x \cdot y) = \chi(\tau_I, x) \cdot \chi(\tau_I, y)$	TI3
A4	$(x + y) \cdot z = x \cdot z + y \cdot z$		
A5	$(x \cdot y) \cdot z = x \cdot (y \cdot z)$	$\tau \mid x = \delta$	TC1
A6	$x + \delta = x$	$x \mid \tau = \delta$	TC2
A7	$\delta \cdot x = \delta$	$(\tau \cdot x) \mid y = x \mid y$	TC3
		$x \mid (\tau \cdot y) = x \mid y$	TC4
C1	$a \mid b = b \mid a$		
C2	$(a \mid b) \mid c = a \mid (b \mid c)$	$x \cdot \tau = x$	T1
C3	$\delta \mid a = \delta$	$\tau \cdot x + x = \tau \cdot x$	T2
		$a \cdot (\tau \cdot x + y) = a \cdot (\tau \cdot x + y) + a \cdot x$	T3
CM1	$x \parallel y = x \mathbin{\lfloor\!\lfloor} y + y \mathbin{\lfloor\!\lfloor} x + x \mid y$		
CM2	$a \mathbin{\lfloor\!\lfloor} x = a \cdot x$	$\tau \mathbin{\lfloor\!\lfloor} x = \tau \cdot x$	TM1
CM3	$(a \cdot x) \mathbin{\lfloor\!\lfloor} y = a \cdot (x \parallel y)$	$(\tau \cdot x) \mathbin{\lfloor\!\lfloor} y = \tau \cdot (x \parallel y)$	TM2
CM4	$(x + y) \mathbin{\lfloor\!\lfloor} z = x \mathbin{\lfloor\!\lfloor} z + y \mathbin{\lfloor\!\lfloor} z$		
CM5	$(a \cdot x) \mid b = (a \mid b) \cdot x$	$\chi(f \circ g, x) = \chi\big(f, \chi(g, x)\big)$	XC1
CM6	$a \mid (b \cdot x) = (a \mid b) \cdot x$	$\chi\big((f \circ g) \circ h, x\big) = \chi\big(f \circ (g \circ h), x\big)$	XC2
CM7	$(a \cdot x) \mid (b \cdot y) = (a \mid b) \cdot (x \parallel y)$		
CM8	$(x + y) \mid z = x \mid z + y \mid z$	$\chi(f, \gamma) = \gamma$	X1
CM9	$x \mid (y + z) = x \mid y + x \mid z$	$\chi(f, \gamma \cdot x) = \gamma \cdot \chi(f, x)$	X2
		$\chi(f, x + y) = \chi(f, x) + \chi(f, y)$	X3
D1	$\chi(\partial_H, a) = a, \text{ if } a \notin H$		
D2	$\chi(\partial_H, a) = \delta, \text{ if } a \in H$	$\chi(\pi^n_J, a) = a$	GPR1
D3	$\chi(\partial_H, x \cdot y) = \chi(\partial_H, x) \cdot \chi(\partial_H, y)$	$\chi(\pi^1_J, a \cdot x) = a, \text{ if } a \in J$	GPR2
		$\chi(\pi^{n+1}_J, a \cdot x) = a \cdot \chi(\pi^n_J, x), \text{ if } a \in J$	GPR3
ID	$\chi(id, x) = x$	$\chi(\pi^n_J, a \cdot x) = a \cdot \chi(\pi^n_J, x), \text{ if } a \notin J$	GPR4

Table 4.1. $ACP_{\tau,u}$.

Definition (4.2.2) *Linear Functional Specifications*

Let N be a finite set of function names. A linear functional specification $E(N)$ for the set of names N is a set of equations of the following form:

$$E(N) = \{\, r_{n,a} | n \in N, a \in A \,\} \cup \{\, e_{n,a} | n \in N, a \in A \,\},$$

in which $r_{n,a}$ is called a boundary condition and $e_{n,a}$ is called a (linear) functional equation. Fix $a \in A$ and $n \in N$. Then we will define what the form is of these two types of equations. First we define the boundary condition. There is a $b \in A \cup \{\delta, \tau\}$ such that

$$r_{n,a} \equiv \chi(n, a) = b.$$

Now we will define what the functional equation is. It can have two forms. We will use the same b as in the boundary condition that we defined above.

$$e_{n,a} \equiv \begin{cases} \chi(n, a \cdot x) = b, & \text{or} \\ \chi(n, a \cdot x) = b \cdot \chi(m, x), & \text{for one } m \in N. \end{cases}$$

In chapter 3 we have only the second form for the functional equation.

Subsequently, we will recall *ODP* and *OSP*. They are the same as in chapter 3 but they use the more general definition of a linear functional specification given above.

Definition (4.2.3)

We will call a mapping $\varphi : N \longrightarrow F$ a *valuation* for N.

Principle (4.2.4) *The Operator Definition Principle*

Let $E(N)$ be a linear functional specification for a set of function names N in the sense of definition (4.2.2). Then the following holds: there is a valuation φ for N, which solves the system of equations $E(N)$. We will use the compact notation *ODP* for this principle.

Principle (4.2.5) *The Operator Specification Principle*

Let $E(N)$ be a linear functional specification for a set of function names N in the sense of definition (4.2.2). Then there is at most one valuation φ for N such that φ solves the system of equations $E(N)$. We will use the compact notation *OSP*.

We will recall *RDP* and *RSP*.

Principle (4.2.6) *The Recursive Definition Principle*

Let E be a guarded recursive specification in the sense of definition (3.2.15). Then there is a solution for E.

Principle (4.2.7) *The Recursive Specification Principle*

Let E be a guarded recursive specification in the sense of definition (3.2.15). Then there is at most one solution for E.

Now we will state our new principle.

Principle (4.2.8) *The Generalized Approximation Induction Principle*

Let $J \subseteq A$ and let $x, y \in P$. If we have for all $n \geq 1 : \pi_J^n(x) = \pi_J^n(y)$ and we have that x or y can be specified with the aid of a guarded recursive specification in the sense of definition (3.2.15), we have $x = y$. We will use the abbreviation *GIP* for this principle; this acronym stands for: "generalized induction principle".

Remarks (4.2.9)

We will explain why this principle is called "generalized". In section 3.2 we can find this principle for $J = A$. The projection operators are simply denoted by π_n there. It has the subsequent form: if we have for all $n \geq 1$:

$$\pi_n(x) = \pi_n(y)$$

then we have $x = y$. This principle is called "the approximation induction principle" hence the adjective generalized. We will also use the notation $\pi_n = \pi_A^n$ in accordance with the notational conventions of process algebra.

Now we will recall *KFAR*. It can be found in chapter 1.

Principle (4.2.10) *Koomen's Fair Abstraction Rule*

Let $x_1, \ldots, x_n$ and $y_1, \ldots, y_n$ be in P. Let $I \subseteq A$ and suppose that we have the following identities for these processes:

$$\begin{aligned} x_1 &= i_1 \cdot x_2 + y_1, \\ x_2 &= i_2 \cdot x_3 + y_2, \\ &\vdots \\ x_{n-1} &= i_{n-1} \cdot x_n + y_{n-1}, \\ x_n &= i_n \cdot x_1 + y_n, \end{aligned}$$

with the following assumptions on the i_j, $(1 \leq j \leq n)$:

$$\{\tau\} \neq \{i_1, \ldots, i_n\} \subseteq I \cup \{\tau\},$$

then we have:

$$\tau_I(x) = \tau \cdot \big(\tau_I(y_1) + \tau_I(y_2) + \cdots + \tau_I(y_n)\big).$$

We will refer to this principle with the abbreviation $KFAR_n$. But if we have that $n = 1$ we will also use the abbreviation *KFAR*.

4.3. Theorems

In this section we will prove some general theorems on linear unary operators. We will focus on theorems in which the projection operators occur. Theorems concerning just linear unary operators without an exit possibility can be found in chapter 3.

The first result is that the projection operators commute. The second result is that in certain circumstances the composition of two projections can be written as one projection. This is true in general for full projections (which is the next result). Then a number of results follow in which necessary conditions are formulated such that a linear unary operator without an exit possibility commutes with a projection operator. This will be generalized to linear unary operators with an exit. The last theorem states that a linear unary operator with an exit possibility can be decomposed into a linear unary operator without an exit and a generalized projection operator. This is an important result. In chapter 3 we discussed the linear unary operators without an exit in depth and in this section we will discuss the projections. This theorem states that we can investigate linear unary operators as they are defined here by researching two simpler classes: the ones in chapter 3 and the generalized projections.

Theorem (4.3.1)
For all $n, k \geq 1$ and $J_1, J_2 \subseteq A$ we have $\pi^n_{J_1} \circ \pi^k_{J_2} = \pi^k_{J_2} \circ \pi^n_{J_1}$.
Proof. Let $M = \{m_{i,j} : 1 \leq i \leq n, 1 \leq j \leq k\}$ be a set of function names. Consider the following linear functional specification:

$$\begin{aligned} E(M) = &\left\{ m_{i,j}(a) = a : a \in A, 1 \leq i \leq n, 1 \leq j \leq k \right\} \\ &\cup \left\{ m_{i,j}(a \cdot x) = m_{i,j}(a) \cdot m_{i,j}(x) \right. \\ &\qquad \left. : a \in A \setminus (J_1 \cup J_2), 1 \leq i \leq n, 1 \leq j \leq k \right\} \\ &\cup \left\{ m_{1,j}(a \cdot x) = m_{1,j}(a) : a \in J_1, 1 \leq j \leq k \right\} \\ &\cup \left\{ m_{i,1}(a \cdot x) = m_{i,1}(a) : a \in J_2, 1 \leq i \leq n \right\} \\ &\cup \left\{ m_{i+1,j}(a \cdot x) = m_{i+1,j}(a) \cdot m_{i,j}(x) \right. \\ &\qquad \left. : a \in J_1 \setminus J_2, 1 \leq i < n, 1 \leq j \leq k \right\} \\ &\cup \left\{ m_{i,j+1}(a \cdot x) = m_{i,j+1}(a) \cdot m_{i,j}(x) \right. \\ &\qquad \left. : a \in J_2 \setminus J_1, 1 \leq i \leq n, 1 \leq j < k \right\} \\ &\cup \left\{ m_{i+1,j+1}(a \cdot x) = m_{i+1,j+1}(a) \cdot m_{i,j}(x) \right. \\ &\qquad \left. : a \in J_1 \cap J_2 1 \leq i < n, 1 \leq j < k \right\}. \end{aligned}$$

We will try the solution $m_{i,j} = \pi^i_{J_1} \circ \pi^j_{J_2}$. First, we will verify that the boundary conditions hold. Because of GPR1 we immediately see that $\pi^i_{J_1} \circ \pi^j_{J_2}(a) = a$ for all $a \in A$ and $1 \leq i \leq n, 1 \leq j \leq k$. Subsequently, we will check that the functional equations hold. Let $a \in A \setminus (J_1 \cup J_2)$ and let $1 \leq i \leq n, 1 \leq j \leq k$ be chosen. Then we see, using GPR4 twice, that

$$\pi^i_{J_1} \circ \pi^j_{J_2}(a \cdot x) = \pi^i_{J_1} \circ \pi^j_{J_2}(a) \cdot \pi^i_{J_1} \circ \pi^j_{J_2}(x).$$

Now we treat the case $a \in J_1, 1 \leq j \leq k$. In fact, we have two subcases here: $a \in J_2$ and $a \notin J_2$. But eventually the remaining part will be "cut off" by $\pi^1_{J_1}$. So we obtain that

$$\pi^1_{J_1} \circ \pi^j_{J_2}(a \cdot x) = \pi^i_{J_1} \circ \pi^j_{J_2}(a).$$

The case $a \in J_2, 1 \leq i \leq n$ is treated similarly to this case only more simply. Now we take $a \in J_1 \setminus J_2$ and $1 \leq i < n, 1 \leq j \leq k$. We find with the aid of the axioms for the projection operators at once that

$$\pi^{i+1}_{J_1} \circ \pi^j_{J_2}(a \cdot x) = \pi^{i+1}_{J_1} \circ \pi^j_{J_2}(a) \cdot \pi^i_{J_1} \circ \pi^j_{J_2}(x).$$

The case $a \in J_2 \setminus J_1, 1 \leq i \leq n, 1 \leq j < k$ is verified the same as the former case. Finally, we will treat the case $a \in J_1 \cap J_2, 1 \leq i < n, 1 \leq j < k$. We find immediately that

$$\pi^{i+1}_{J_1} \circ \pi^{j+1}_{J_2}(a \cdot x) = \pi^{i+1}_{J_1} \circ \pi^{j+1}_{J_2}(a) \cdot \pi^i_{J_1} \circ \pi^j_{J_2}(x).$$

Now we can conclude that the solution that we proposed is indeed a solution for $E(M)$. On the other hand, if we try the solution $m_{i,j} = \pi^j_{J_2} \circ \pi^i_{J_1}$, we obtain, just as above, that this will solve the system $E(M)$, too. Thus, we may conclude in accord with *OSP* that these solutions are the same. In particular, we find that $\pi^n_{J_1} \circ \pi^k_{J_2} = \pi^k_{J_2} \circ \pi^n_{J_1}$. Herewith we end the verification of 4.3.1.

Theorem (4.3.2)

For all $k \geq n \geq 1$ and for all subsets $J_1, J_2 \subseteq A$ with $J_2 \subseteq J_1$ we have

$$\pi^n_{J_1} \circ \pi^k_{J_2} = \pi^n_{J_1}.$$

Proof. We will prove 4.3.2 with induction on n. Hence, let $n = 1, k \geq 1$ and $J_2 \subseteq J_1 \subseteq A$ be chosen. Let m be a function name. Consider the linear functional specification $E(m)$, defining $\pi^1_{J_1}$, hereinafter.

$$\begin{aligned} E(m) = \{\, & m(a) = a : a \in A \,\} \\ & \cup \{\, m(a \cdot x) = m(a) : a \in J_1 \,\} \\ & \cup \{\, m(a \cdot x) = m(a) \cdot m(x) : a \in A \setminus J_1 \,\}. \end{aligned}$$

We immediately see that $\pi^1_{J_1}$ is a solution for $E(m)$. We will show that $\pi^1_{J_1} \circ \pi^k_{J_2}$ is also a solution. Thereto we will verify that the boundary conditions hold.

Let $a \in A$ be fixed. Then we see with GPR1 that $\pi^1_{J_1} \circ \pi^k_{J_2}(a) = a$. Now we will consider the functional equations. First let $a \in J_1$. We will have two cases: $a \notin J_2$ and $a \in J_2$. The latter case will have two subcases: $k = 1$ and $k > 1$. Consider the following simple calculation:

$$\pi^1_{J_1} \circ \pi^k_{J_2}(a \cdot x) = \begin{cases} a \in J_2 : & \begin{cases} \pi^1_{J_1} \circ \pi^k_{J_2}(a), & \text{if } k = 1; \\ \pi^1_{J_1}\big(\pi^k_{J_2}(a) \cdot \pi^{k-1}_{J_2}(x)\big), & \text{if } k > 1. \end{cases} \\ & \\ a \notin J_2 : & \pi^1_{J_1}\big(\pi^k_{J_2}(a) \cdot \pi^k_{J_2}(x)\big) \end{cases}$$

$$= \pi^1_{J_1} \circ \pi^k_{J_2}(a).$$

Now we will treat the case $a \in A \setminus J_1 \subseteq A \setminus J_2$. It is evident that

$$\pi^1_{J_1} \circ \pi^k_{J_2}(a \cdot x) = \pi^1_{J_1} \circ \pi^k_{J_2}(a) \cdot \pi^1_{J_1} \circ \pi^k_{J_2}(x).$$

Thus, we find, using *OSP*:

$$\forall k \geq 1,\, \forall J_2 \subseteq J_1 : \pi^1_{J_1} \circ \pi^k_{J_2} = \pi^1_{J_1}. \tag{1}$$

Now suppose that 4.3.2 is proved up to n inclusive. Then we will verify 4.3.2 for $n+1$. Choose $k \geq n+1$ and let $J_2 \subseteq J_1 \subseteq A$ be fixed. Let

$$M = \{m_1, \ldots, m_{n+1}\}$$

be a set of function names and consider the subsequent linear functional specification $E(M)$:

$$\begin{aligned} E(M) = \big\{\, & m_i(a) = a : a \in A,\, 1 \leq i \leq n+1 \,\big\} \\ & \cup \big\{\, m_i(a \cdot x) = m_i(a) \cdot m_i(x) : a \in A \setminus J_1,\, 1 \leq i \leq n+1 \,\big\} \\ & \cup \big\{\, m_{i+1}(a \cdot x) = m_{i+1}(a) \cdot m_i(x) : a \in J_1,\, 1 \leq i \leq n \,\big\} \\ & \cup \big\{\, m_1(a \cdot x) = m_1(a) : a \in J_1 \,\big\}. \end{aligned}$$

It will be clear that $\pi^1_{J_1}, \ldots, \pi^{n+1}_{J_1}$ solves the system $E(M)$. We will try the solution $m_i = \pi^i_{J_1} \circ \pi^k_{J_2}$ for $1 \leq i \leq n+1$. It is obvious that the boundary conditions are satisfied. So we will only treat the functional equations. Let thereto $a \in A \setminus J_1$ and $1 \leq i \leq n+1$. Since $a \notin J_2$ we find:

$$\pi^i_{J_1} \circ \pi^k_{J_2}(a \cdot x) = \pi^i_{J_1} \circ \pi^k_{J_2}(a) \cdot \pi^i_{J_1} \circ \pi^k_{J_2}(x).$$

Now let $a \in J_1$. First we will handle the case $i = 1$.

$$\begin{aligned} \pi^1_{J_1} \circ \pi^k_{J_2}(a \cdot x) &= \pi^1_{J_1}(a \cdot x) && \text{because of (1)} \\ &= \pi^1_{J_1}(a) \\ &= \pi^1_{J_1} \circ \pi^k_{J_2}(a). \end{aligned}$$

Now let $1 \leq i < n$. Consider the following calculation.

$$\begin{aligned}
\pi_{J_1}^{i+1} \circ \pi_{J_2}^{k}(a \cdot x) &= \pi_{J_1}^{i+1}(a \cdot x) && \text{induction} \\
&= \pi_{J_1}^{i+1}(a) \cdot \pi_{J_1}^{i}(x) \\
&= \pi_{J_1}^{i+1} \circ \pi_{J_2}^{k}(a) \cdot \pi_{J_1}^{i}(x) \\
&= \pi_{J_1}^{i+1} \circ \pi_{J_2}^{k}(a) \cdot \pi_{J_1}^{i} \circ \pi_{J_2}^{k}(x). && \text{induction}
\end{aligned}$$

Now we will treat the case $i = n + 1$. We will have two subcases: $a \in J_2$ and $a \notin J_2$. First, we will consider $a \in J_2$ (observe that $k > 1$):

$$\begin{aligned}
\pi_{J_1}^{n+1} \circ \pi_{J_2}^{k}(a \cdot x) &= \pi_{J_1}^{n+1}\big(\pi_{J_2}^{k}(a) \cdot \pi_{J_2}^{k-1}(x)\big) \\
&= \pi_{J_1}^{n+1} \circ \pi_{J_2}^{k}(a) \cdot \pi_{J_1}^{n} \circ \pi_{J_2}^{k-1}(x) \\
&= \pi_{J_1}^{n+1} \circ \pi_{J_2}^{k}(a) \cdot \pi_{J_1}^{n}(x) && \text{induction} \\
&= \pi_{J_1}^{n+1} \circ \pi_{J_2}^{k}(a) \cdot \pi_{J_1}^{n} \circ \pi_{J_2}^{k}(x). && \text{induction}
\end{aligned}$$

Now we take $a \notin J_2$.

$$\begin{aligned}
\pi_{J_1}^{n+1} \circ \pi_{J_2}^{k}(a \cdot x) &= \pi_{J_1}^{n+1}\big(\pi_{J_2}^{k}(a) \cdot \pi_{J_2}^{k}(x)\big) \\
&= \pi_{J_1}^{n+1} \circ \pi_{J_2}^{k}(a) \cdot \pi_{J_1}^{n} \circ \pi_{J_2}^{k}(x).
\end{aligned}$$

We see that in both subcases the right-hand sides are the same. Hence, we see that the functional equations are satisfied, too. So with the aid of *OSP* we find that $\pi_{J_1}^{n+1} \circ \pi_{J_2}^{k} = \pi_{J_1}^{n+1}$. This will end the induction step and therewith the proof of 4.3.2.

Corollary (4.3.3)

For all $n, k \geq 1$ and for all $J \subseteq A$ we have $\pi_J^n \circ \pi_J^k = \pi_J^{\min(n,k)}$.

Proof. Let $J_1 = J_2 = J$. First suppose that $k \geq n$. With the aid of theorem (4.3.2) we find that

$$\pi_J^n \circ \pi_J^k = \pi_J^n = \pi_J^{\min(n,k)}.$$

Now suppose that $k < n$. Then we find:

$$\begin{aligned}
\pi_J^n \circ \pi_J^k &= \pi_J^k \circ \pi_J^n && \text{because of (4.3.1)} \\
&= \pi_J^k && \text{theorem (4.3.2)} \\
&= \pi_J^{\min(n,k)}.
\end{aligned}$$

This will end the proof of 4.3.3.

Corollary (4.3.4)

For all $n, k \geq 1 : \pi_n \circ \pi_k = \pi_{\min(n,k)}$. We used here the abbreviation $\pi_n = \pi_A^n$, see (4.2.9).

Remark (4.3.5)

It is a well-known fact in process algebra that for closed terms t the following holds:

$$\pi_n \circ \pi_k(t) = \pi_{\min(n,k)}(t).$$

Confer section 2.4. We see that with the use of linear functional specifications, in combination with *ODP* and *OSP*, we are able to prove this fact for all processes. So a composition of "full" projections is always composable into only one full projection. This composability result is generalized in theorem (4.3.2). The following question may arise: "Is every composition of generalized projection operators composable into only one (generalized) projection operator?" The answer to this question is negative. For, let $A = \{a, b\}$ and suppose that there exists a subset $J \subseteq A$ and a natural number $n \geq 1$ such that $\pi^1_{\{a\}} \circ \pi^2_{\{b\}} = \pi^n_J$. Assume that $b \notin J$. Then we see that

$$\pi^1_{\{a\}} \circ \pi^2_{\{b\}}(b^3) = b^2 \neq b^3 = \pi^n_J(b^3).$$

So this assumption cannot hold. Hence, $b \in J$. In a similar way we find that $a \in J$, so $J = A$. Assume that $n = 1$. Then we see that

$$\pi^1_J(b^3) = b \neq b^2 = \pi^1_{\{a\}} \circ \pi^2_{\{b\}}(b^3).$$

Thus, we find that $n > 1$. Thus, $\pi^n_J(a^2) = a^2$. But

$$\pi^n_J(a^2) = \pi^1_{\{a\}} \circ \pi^2_{\{b\}}(a^2) = a.$$

So $n \not> 1$. This is in contradiction with the fact that $n > 1$ so the assumption that $\pi^1_{\{a\}} \circ \pi^2_{\{b\}}$ can be written as π^n_J for one $n \geq 1$ and a subset $J \subseteq A$ cannot hold.

We will define hereinafter what a renaming operator is. In chapter 3 we define this to be a linear unary operator that can be defined with the aid of a linear functional specification with the extra requirement that the set of derived operators $D(f)$ contains only one element; see definition (3.2.10). But with the generalized version of a linear functional specification, we are able to specify π_1. This means that with this definition π_1 would become a renaming operator. To solve this, we will need the subsequent definition in which we will introduce the concept of the set of "exits" of a linear unary operator.

Definition (4.3.6)

Let $f \in F$ be a linear unary operator. The set of exits $X(f)$ with respect to f is the following subset of A:

$$X(f) = \{\, a \in A \mid \forall x \in P \exists b \in A \cup \{\tau\} : f(a \cdot x) = b \,\}.$$

We excluded δ here, since $\delta = \delta \cdot f(x)$ for all $f \in F$. This means that if we have an atomic action a such that $f(a \cdot x) = \delta$, it looks like an exit but, in fact, it is not since we can write $f(a \cdot x) = \delta \cdot f(x)$.

Examples (4.3.7)

It is easy to see that $X(\pi_J^1) = J$. For, let $j \in J$ then we have for all $x \in P$:

$$\pi_J^1(j \cdot x) = j$$

so we see that $j \in X(\pi_J^1)$. Now suppose that $a \notin J$. Then we know that for all $x \in P : \pi_J^1(a \cdot x) = a \cdot \pi_J^1(x)$ and we see that $a \notin X(\pi_J^1)$. Thus, we find that $X(\pi_J^1) = J$. We will show that $X(\partial_H) = \emptyset$. Suppose that there is an atomic action $a \in X(\partial_H)$. If $a \notin H$ we find that $\partial_H(a \cdot x) = a \cdot \partial_H(x)$. This yields that $a \notin X(\partial_H)$. So we must have $a \in H$. But then we find: $\partial_H(a \cdot x) = \delta$. So we see, by definition, that $a \notin X(\partial_H)$. We find thus that $X(\partial_H) = \emptyset$. In a similar way we can obtain that $X(\tau_I) = \emptyset$.

Definition (4.3.8)

Let $f \in F$ be a linear unary operator, which can be defined with the aid of a linear functional specification. If $|D(f)| = 1$ and $X(f) = \emptyset$, we will call f a (linear unary) renaming operator.

Examples (4.3.9)

We will show that ∂_H is a renaming operator. We have seen in (3.2.9) that $|D(\partial_H)| = 1$. In (4.3.7) we have seen that $X(\partial_H) = \emptyset$. It is easy to see that ∂_H can be defined with the aid of a linear functional specification, so we find that ∂_H is a renaming operator. In the same way we can obtain that τ_I is a renaming operator. Now suppose that π_J^1 is a renaming operator, then π_J^1 must be the identity map. We already know that $|D(\pi_J^1)| = 1$; see (3.2.9). Since π_J^1 is a renaming operator, we have $J = X(\pi_J^1) = \emptyset$. Let n be a function name and consider the following linear functional specification.

$$E(n) = \{\, n(a) = a : a \in A \,\} \cup \{\, n(a \cdot x) = n(a) \cdot n(x) : a \in A \,\}.$$

It is very easy to see that both $\pi_\emptyset^1$ and id* are solutions for $E(n)$. Hence, we have $\pi_\emptyset^1 = id$.

* See table 4.1 page 155.

Theorem (4.3.10)

Let $f \in F$ be a renaming operator. Let $J \subseteq A$ be chosen. Suppose that the following two conditions are valid

(i) $\quad f(J) \subseteq J \cup \{\delta\}$

(ii) $\quad f(A \setminus J) \subseteq A \setminus J \cup \{\delta\}$

then we have for all $n \geq 1 : f \circ \pi_J^n = \pi_J^n \circ f$. Observe that f is a concrete renaming operator because of the conditions (i) and (ii).

Proof. Choose an $n \geq 1$ and let $M = \{m_1, \ldots, m_n\}$ be a set of function names. Contemplate the subsequent linear functional specification.

$$\begin{aligned} E(M) = &\left\{\, m_i(a) = f(a) : a \in A,\, 1 \leq i \leq n \,\right\} \\ &\cup \left\{\, m_i(a \cdot x) = m_i(a) \cdot m_i(x) : a \in A \setminus J,\, 1 \leq i \leq n \,\right\} \\ &\cup \left\{\, m_1(a \cdot x) = m_1(a) : a \in J \,\right\} \\ &\cup \left\{\, m_{i+1}(a \cdot x) = m_{i+1}(a) \cdot m_i(x) : a \in J,\, 1 \leq i < n \,\right\}. \end{aligned}$$

We will try the solution $m_i = \pi_J^i \circ f$ for $1 \leq i \leq n$. It is very easy to see that the boundary conditions are valid, so we will only verify that the functional equations are satisfied. Let $a \in A \setminus J$ and $1 \leq i \leq n$.

$$\begin{aligned} \pi_J^i \circ f(a \cdot x) &= \pi_J^i\big(f(a) \cdot f(x)\big) \\ &= \pi_J^i \circ f(a) \cdot \pi_J^i \circ f(x). && \text{use } (ii) \end{aligned}$$

Now let $a \in J$. We have two cases: $i = 1$ and $n \geq i > 1$.

$$\begin{aligned} \pi_J^1 \circ f(a \cdot x) &= \pi_J^1\big(f(a) \cdot f(x)\big) \\ &= \pi_J^1 \circ f(a). && \text{use } (i) \end{aligned}$$

Now the case $i > 1$:

$$\begin{aligned} \pi_J^{i+1} \circ f(a \cdot x) &= \pi_J^{i+1}\big(f(a) \cdot f(x)\big) \\ &= \pi_J^{i+1} \circ f(a) \cdot \pi_J^i \circ f(x). && \text{use } (i) \end{aligned}$$

So we see that this solves the system $E(M)$. On the other hand if we try the solution $m_i = f \circ \pi_J^i$ for all $1 \leq i \leq n$, we obtain that this is also a solution for $E(M)$. Using *OSP* we find that, in particular, $f \circ \pi_J^n = \pi_J^n \circ f$. This will end the verification of the theorem.

Theorem (4.3.10) was originally stated in chapter 3. In there, it was not yet possible to reason with projection operators. We obtained that $f \circ \pi_n(x) = \pi_n \circ f(x)$ for so-called concrete processes x; see theorem (3.4.8).

Corollary (4.3.11)

For all $n \geq 1$ and for all subsets $H, J \subseteq A$ we have:

$$\pi_J^n \circ \partial_H = \partial_H \circ \pi_J^n.$$

Proof. We will check that the conditions of theorem (4.3.10) are satisfied. First, we will take a look at condition (i). Since $\partial_H(a) \subseteq \{a, \delta\}$ for all $a \in A$, we immediately see that (i) holds. The same is valid for condition (ii). Thus, we may apply theorem (4.3.10) and we obtain the desired equation.

Theorem (4.3.12)

Let $I, J \subseteq A$ and suppose that $I \cap J = \emptyset$. Then we have for all $n \geq 1$: $\pi_J^n \circ \tau_I = \tau_I \circ \pi_J^n$.

Proof. Let $n \geq 1$ be chosen. Let $\{m_1, \ldots, m_n\}$ be a set of function names. Consider the following linear functional specification:

$$\begin{aligned} E(m_1, \ldots, m_n) = & \left\{ m_i(a) = a : a \in A \setminus I,\, 1 \leq i \leq n \right\} \\ & \cup \left\{ m_i(a) = \tau : a \in I,\, 1 \leq i \leq n \right\} \\ & \cup \left\{ m_i(a \cdot x) = m_i(a) \cdot m_i(x) : a \in A \setminus J,\, 1 \leq i \leq n \right\} \\ & \cup \left\{ m_1(a \cdot x) = m_1(a) : a \in J \right\} \\ & \cup \left\{ m_{i+1}(a \cdot x) = m_{i+1}(a) \cdot m_i(x) : a \in J,\, 1 \leq i < n \right\}. \end{aligned}$$

It consists of straightforward calculation to see that if we put for all $1 \leq i \leq n$:

$$m_i = \pi_J^i \circ \tau_I,$$

that this solves the system $E(m_1, \ldots, m_n)$. But on the other hand if we put

$$m_i = \tau_I \circ \pi_J^i,$$

for all $1 \leq i \leq n$, we obtain a solution for $E(m_1, \ldots, m_n)$, too. So with the aid of *OSP* we may conclude that these solutions are the same. In particular we find that $\pi_J^n \circ \tau_I = \tau_I \circ \pi_J^n$. This will end the verification of 4.3.12.

Theorem (4.3.13)

Let f be a renaming operator. Let $J \subseteq A$. Suppose that the following conditions hold.

(i) $\quad f(J) \subseteq J \cup \{\delta\}$

(ii) $\quad f(A \setminus J) \subseteq A \setminus J \cup \{\delta, \tau\}$

Then we have $\pi_J^n \circ f = f \circ \pi_J^n$ for all $n \geq 1$.

Proof. We could give a direct proof of this theorem, but we will give an indirect proof: first, we will show that we can decompose f into a concrete part and an abstraction operator. Thereinafter, we will apply theorems (4.3.10) and (4.3.12). Let $I = \{a \in A : f(a) = \tau\}$ be the *abstracting set* of f. Let n be a function name and consider the following linear functional specification.

$$\begin{aligned} E(n) = &\left\{ n(a) = f(a) : a \in A \setminus I \right\} \\ &\cup \left\{ n(a) = a : a \in I \right\} \\ &\cup \left\{ n(a \cdot x) = n(a) \cdot n(x) : a \in A \right\}. \end{aligned}$$

It will be clear that the solution for this linear functional specification is a concrete renaming operator. We will name it g. It is not very difficult to see that it will be sufficient to show that

$$\forall a \in A : f(a) = g \circ \tau_I(a), \tag{2}$$

in order to be able to conclude that $f = g \circ \tau_I$. Since (2) is valid, we find the desired decomposition. We will verify that g satisfies the conditions (i) and (ii) of theorem (4.3.10). Let $a \in J$, then we see that $a \notin I$*. So, we find $g(a) = f(a) \in J \cup \{\delta\}$. Now we verify condition (ii). Let $a \in A \setminus J$. If $a \in I$, we find, by definition, $g(a) = a \in A \setminus J \cup \{\delta\}$. So let $a \notin I$. Then we find $g(a) = f(a) \in A \setminus J \cup \{\delta, \tau\}$. But $g(a) \neq \tau$, so $g(a) \in A \setminus J \cup \{\delta\}$. It will be clear that we may use theorem (4.3.12), so we can make the subsequent calculation.

$$\begin{aligned} f \circ \pi_J^n &= g \circ \tau_I \circ \pi_J^n \\ &= g \circ \pi_J^n \circ \tau_I && \text{because of (4.3.12)} \\ &= \pi_J^n \circ g \circ \tau_I && \text{because of (4.3.10)} \\ &= \pi_J^n \circ f. \end{aligned}$$

This will end the verification of 4.3.13.

Remark (4.3.14)

The decomposition of a renaming operator f into a concrete renaming operator g and an abstraction operator τ_I is not unique. For, let $A = \{a, b\}$ and let $I = \{b\}$. Let f be the solution for $E(n)$:

$$E(n) = \left\{ n(a) = b, n(b) = \tau \right\} \cup \left\{ n(c \cdot x) = n(c) \cdot n(x) : c \in A \right\}.$$

Let g be the solution for $E(m)$:

$$E(m) = \left\{ m(a) = b, m(b) = b \right\} \cup \left\{ m(c \cdot x) = m(c) \cdot m(x) : c \in A \right\}.$$

---

* We see that $I \cap J = \emptyset$, so the condition of theorem (4.3.12) is satisfied

Then it is easy to see that $f = g \circ \tau_I$ (of course, g is a concrete renaming operator). Let g' be the solution for $E(m')$:

$$E(m') = \{\, m'(a) = b, m'(b) = a \,\} \cup \{\, m'(c \cdot x) = m'(c) \cdot m'(x) : c \in A \,\}.$$

We can find easily that $f = g' \circ \tau_I$. Suppose that $g = g'$ then g' is also a solution for $E(m)$. But $g'(b) \neq b$ and we find that $g' \neq g$. So the decomposition is *not* unique.

Theorem (4.3.15)

Let $f_1 \in F$ be a linear unary operator that can be defined with the aid of a linear functional specification. Let $D(f_1) = \{f_1, \ldots, f_k\}$ be the set of derived operators of f_1. Let $X(f_j) = \emptyset$ for all $1 \leq j \leq k$. Suppose that the following conditions hold.

(*i*) $\bigcup_{j=1}^{k} f_j(J) \subseteq J \cup \{\delta\}$

(*ii*) $\bigcup_{j=1}^{k} f_j(A \setminus J) \subseteq A \setminus J \cup \{\delta, \tau\}$

Then we have $\pi_J^n \circ f_1 = f_1 \circ \pi_J^n$ for all $n \geq 1$.

Proof. Define a map $\sigma : A \times \{1, 2, \ldots, k\} \longrightarrow \{1, 2, \ldots, k\}$ as follows

$$\sigma(a, i) = j \iff \forall x \in P : f_i(a \cdot x) = f_i(a) \cdot f_j(x).$$

Let $M = \{m_{i,j} : 1 \leq i \leq n,\ 1 \leq j \leq k\}$ be a set of function names and consider the subsequent linear functional specification $E(M)$.

$$\begin{aligned} E(M) = \{\, & m_{i,j}(a) = f_j(a) : a \in A,\ 1 \leq i \leq n,\ 1 \leq j \leq k \,\} \\ & \cup \{\, m_{i,j}(a \cdot x) = m_{i,j}(a) \cdot m_{i,\sigma(a,j)}(x) \\ & \qquad : a \in A \setminus J,\ 1 \leq i \leq n,\ 1 \leq j \leq k \,\} \\ & \cup \{\, m_{1,j}(a \cdot x) = m_{1,j}(a) : a \in J,\ 1 \leq j \leq k \,\} \\ & \cup \{\, m_{i+1,j}(a \cdot x) = m_{i+1,j}(a) \cdot m_{i,\sigma(a,j)}(x) \\ & \qquad : a \in J,\ 1 \leq i < n,\ 1 \leq j \leq k \,\}. \end{aligned}$$

It is very easy to see that if we try the solution $\pi_J^i \circ f_j = m_{i,j}$ this system will solve the linear functional specification $E(M)$. We will need here both conditions. But if we try on the other hand the solution $f_j \circ \pi_J^i = m_{i,j}$ we will effortlessly see that this will solve the linear functional specification, too. So in accord with *OSP* we may conclude that these solutions must be equal. In particular, we find that $\pi_J^n \circ f_1 = f_1 \circ \pi_J^n$ for all $n \geq 1$. This will end the proof of the theorem.

Below, we will generalize theorem (4.3.15) to linear unary operators that can have an exit.

Theorem (4.3.16)

Let $f_1 \in F$ be a linear unary operator that can be defined with the aid of a linear functional specification. Let $D(f_1) = \{f_1, \dots, f_k\}$ be its set of derived operators. Suppose that the following conditions hold.

(i) $\bigcup_{j=1}^{k} f_j\big(J \setminus X(f_j)\big) \subseteq J \cup \{\delta\}$

(ii) $\bigcup_{j=1}^{k} f_j\big(A \setminus \big(J \cup X(f_j)\big)\big) \subseteq A \setminus J \cup \{\delta, \tau\}$

Then we have $\pi_J^n \circ f_1 = f_1 \circ \pi_J^n$ for all $n \geq 1$.

Proof. Let $M = \{m_{i,j} : 1 \leq i \leq n,\ 1 \leq j \leq k\}$ be a set of function names. Let

$$\sigma : A \times \{1, 2, \dots, k\} \longrightarrow \{1, 2, \dots, k\}$$

be the partial* function defined hereinafter.

$$\sigma(a, i) = j \iff \forall\, x \in P : f_i(a \cdot x) = f_i(a) \cdot f_j(x).$$

Consider the next linear functional specification $E(M)$.

$$\begin{aligned} E(M) = \big\{\, & m_{i,j}(a) = f_j(a) : 1 \leq i \leq n,\ 1 \leq j \leq k \,\big\} \\ & \cup \bigcup_{j=1}^{k} \big\{\, m_{i,j}(a \cdot x) = m_{i,j}(a) : a \in X(f_j),\ 1 \leq i \leq n \,\big\} \\ & \cup \bigcup_{j=1}^{k} \big\{\, m_{1,j}(a \cdot x) = m_{1,j}(a) : a \in J \setminus X(f_j) \,\big\} \\ & \cup \bigcup_{j=1}^{k} \big\{\, m_{i+1,j}(a \cdot x) = m_{i+1,j}(a) \cdot m_{i,\sigma(a,j)}(x) \\ & \qquad : a \in J \setminus X(f_j),\ 1 \leq i < n \,\big\} \\ & \cup \bigcup_{j=1}^{k} \big\{\, m_{i,j}(a \cdot x) = m_{i,j}(a) \cdot m_{i,\sigma(a,j)}(x) \\ & \qquad : a \in A \setminus \big(J \cup X(f_j)\big),\ 1 \leq i \leq n \,\big\}. \end{aligned}$$

It is rather easy to deduce that, if we put $m_{i,j} = \pi_J^i \circ f_j$, this will solve the linear functional specification $E(M)$ (we will need here the conditions stated in the theorem). But if we put, on the other hand, $m_{i,j} = f_j \circ \pi_J^i$, a simple calculation will learn us that this will solve the system $E(M)$, too. Using *OSP* we immediately see that 4.3.16 is proved.

* If $a \in X(f_i)$ then $\sigma(a, i)$ is not defined.

Definition (4.3.17)

A linear unary operator $f \in F$ that can be defined with the aid of a linear functional specification has no exits if for all $g \in D(f)$ their sets of exits $X(g)$ are empty.

Theorem (4.3.18)

Let $f \in F$ be a linear unary operator that can be defined with the aid of a linear functional specification. Then there is a linear unary operator $g \in F$ without exits such that $f = g \circ \pi^1_{X(f)}$. Moreover, the sets of derived operators of f and g have the same cardinality.

Proof. Let $f = f_1$ and let $D(f) = \{f_1, \ldots, f_k\}$ be the set of derived operators of f. Let $N = \{n_1, \ldots, n_k\}$ be a set of function names. Let

$$\sigma : A \times \{1, 2, \ldots, k\} \longrightarrow \{1, 2, \ldots, k\}$$

be the partial map defined as follows.

$$\sigma(a, i) = j \iff \forall\, x \in P : f_i(a \cdot x) = f_i(a) \cdot f_j(x).$$

Consider the linear functional specification $E(N)$ below.

$$\begin{aligned} E(N) = \big\{\, & n_i(a) = f_i(a) : a \in A,\, 1 \le i \le k \,\big\} \\ & \cup \bigcup_{i=1}^{k} \big\{\, n_i(a \cdot x) = n_i(a) \cdot n_i(x) : a \in X(f_i) \,\big\} \\ & \cup \bigcup_{i=1}^{k} \big\{\, n_i(a \cdot x) = n_i(a) \cdot n_{\sigma(a,i)}(x) : a \in A \setminus X(f_i) \,\big\}. \end{aligned}$$

According to *ODP* there is a solution for this system. Let us say that $g_1, \ldots, g_k$ is a solution for it. We claim that we can take $g = g_1$. For, let

$$M = \{m_1, \ldots, m_k\}$$

be a set of function names. Let $E(M)$ be the defining linear functional specification of f:

$$\begin{aligned} E(M) = \big\{\, & m_i(a) = f_i(a) : a \in A,\, 1 \le i \le k \,\big\} \\ & \cup \bigcup_{i=1}^{k} \big\{\, m_i(a \cdot x) = m_i(a) : a \in X(f_i) \,\big\} \\ & \cup \bigcup_{i=1}^{k} \big\{\, m_i(a \cdot x) = m_i(a) \cdot m_{\sigma(a,i)}(x) : a \in A \setminus X(f_i) \,\big\}. \end{aligned}$$

It will be evident that $f_1, \ldots, f_k$ is a solution for this system. It is also very easy to see that

$$g_1 \circ \pi^1_{X(f_1)}, g_2 \circ \pi^1_{X(f_2)}, \ldots, g_k \circ \pi^1_{X(f_k)}$$

is also a solution for it. So with the aid of *OSP* we can conclude that these solutions are the same. In particular, we find that $f = g_1 \circ \pi^1_{X(f)}$. We see that $|D(f)| = |D(g)|$. This will finish the proof of 4.3.18.

4.4. Applications

In this section we will apply the theory that we developed. First, we will state a correctness criterion that can be found among others in [**13**]. Then we will show that processes that satisfy other criteria that are common in protocol verification* also satisfy the criterion stated in this chapter (we will prove this in two ways). Furthermore, we will show that a FIFO queue with an unbounded capacity satisfies this correctness criterion. We will base ourselves on a (begin of a) proof that can be found in [**13**]. However, as we can see in the preface of [**3**]: the authors of [**13**] have not been able to give a solution to the problems they have put forward. Then we will formulate a lemma on an apparently important linear unary operator that occurs in the proof of the FIFO queue. Afterwards, we will discuss three alternating bit protocols with a time out that can take place at any moment. They only differ in one way: their communication channels. The acknowledgement channel is perfectly reliable in all three cases. This is done to shorten the proofs but is not a serious constraint. In the first protocol the data channel is also reliable: it is a queue with an unbounded capacity, since the time out can occur at any moment. In the second protocol we make things a bit less unrealistic: the queue with an unbounded capacity can lose its last read datum at any moment. In the last protocol we have a queue that can lose or duplicate any datum that is in the queue at any moment.

The following definition is taken from [**13**].

Definition (4.4.1)

Let D be a set of datum elements. Let P be a process (see figure 4.2). Let $H \subseteq A$ be the set of all the internal read and send actions and the input read and send actions of the process P. Let $I \subseteq A$ be the set of all the non-trivial communications that can be made with the atomic actions of H. Let D^ω be the cartesian product of infinitely many countable copies of D. Let the input

* See, for instance, [**17**], [**18**] and [**29**].

and output channel of P be labeled with the numbers 1, and 2 respectively. If for all $(d_i)_i \in D^\omega$:

$$\tau_I \circ \partial_H\Big(P \parallel \prod_{i=1}^{\infty} s_1(d_i)\Big) = \tau \cdot \prod_{i=1}^{\infty} s_2(d_i),$$

then we will call P a correct (communication) protocol.

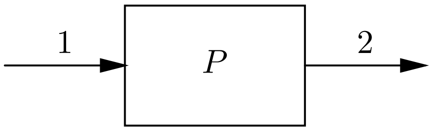

Figure 4.2. Visualization of the process P.

In protocol verifications we often meet other correctness criteria. One of these criteria is that a protocol is correct if it is bisimilar to a one bit buffer. In other words, if P is our protocol with input channel 1 and output channel 2 then we want that

$$P = \sum_{d \in D} r_1(d) \cdot s_2(d) \cdot P.$$

Another criterion says that the protocol should behave as a two bit buffer. And, more general, we have a criterion that says that our protocol is correct when it behaves as an s-bit buffer. The criterion that we use in this chapter says that the protocol must behave like a queue with unbounded capacity (or an "ω-bit buffer").

Hereinafter, we will show that every s-bit buffer satisfies the correctness criterion stated in definition (4.4.1). Consider the following visualization of an s-bit buffer.

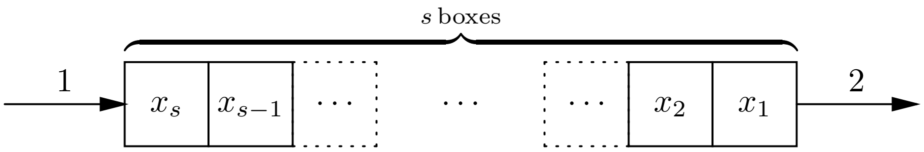

Figure 4.3. An s-bit buffer with input channel 1 and output channel 2.

Now we will give a guarded recursive specification of an s-bit buffer.

$$B_s = \sum_{x_1 \in D} r_1(x_1) \cdot B_s(x_1)$$

For $1 \leq j < s$ we have:

$$B_s(x_1,\dots,x_j) = \sum_{x_{j+1}\in D} r_1(x_{j+1})\cdot B_s(x_1,\dots,x_{j+1}) + s_2(x_1)\cdot B_s(x_2,\dots,x_j)$$
$$B_s(x_1,\dots,x_s) = s_2(x_1)\cdot B_s(x_2,\dots,x_s).$$

Let the following subsets of the set of atomic actions A be given:

$$H = \{\, r_1(x), s_1(x) : x\in D \,\},$$
$$I = \{\, c_1(x) : x\in D \,\}.$$

Below we will prove that B_s is a correct protocol. But first, we will provide a visualization in figure 4.4 of the left-hand side of the following equation (1).

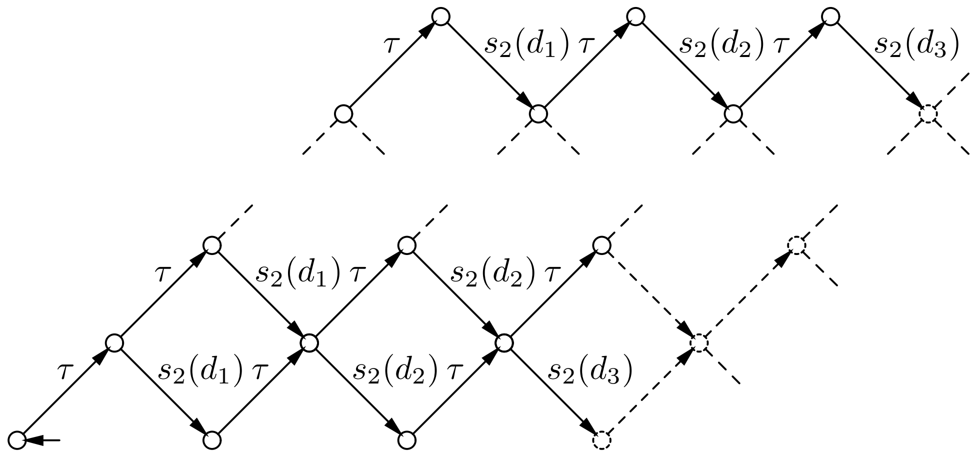

Figure 4.4. A graph of the left-hand side of equation (1).

Theorem (4.4.2)

For all $(d_i)_i \in D^\omega$ the following holds:

$$\tau_I\circ\partial_H\Big(B_s \parallel \prod_{i=1}^{\infty} s_1(d_i)\Big) = \tau\cdot\prod_{i=1}^{\infty} s_2(d_i). \tag{1}$$

Proof. Let us choose an element $(d_i)_i\in D^\omega$. We will introduce some notations for convenience sake. For all $k\ge 1$ and $1\le j\le s$ we define

$$Z_0^k = B_s \parallel \prod_{i=k}^{\infty} s_1(d_i)$$

$$Z_j^k = B_s(d_k,\dots,d_{k+j-1}) \parallel \prod_{i=k+j}^{\infty} s_1(d_i).$$

It is very easy to see that the following holds for all $k \geq 1$ and $1 \leq j < s$:

$$\begin{aligned}
&\tau_I \circ \partial_H(Z_0^k) = \tau \cdot \tau_I \circ \partial_H(Z_1^k) \\
&\tau_I \circ \partial_H(Z_j^k) = \tau \cdot \tau_I \circ \partial_H(Z_{j+1}^k) + s_2(d_k) \cdot \tau_I \circ \partial_H(Z_{j-1}^{k+1}) \\
&\tau_I \circ \partial_H(Z_s^k) = s_2(d_k) \cdot \tau_I \circ \partial_H(Z_{s-1}^{k+1}).
\end{aligned} \tag{2}$$

We will prove the following claim:

$$\forall k, n \geq 1, \forall 0 \leq j \leq s: \tag{3}$$

$$\pi_n \circ \tau_I \circ \partial_H(Z_j^k) = \begin{cases} \tau \cdot \pi_n\Big(\prod_{i=k}^{\infty} s_2(d_i)\Big), & \text{if } 0 \leq j < s; \\ \pi_n\Big(\prod_{i=k}^{\infty} s_2(d_i)\Big), & \text{if } j = s. \end{cases}$$

We will prove (3) with induction on n. So let $n = 1$. Now fix a $k \geq 1$. First, we will take $j = s$. Since

$$\pi_1 \circ \tau_I \circ \partial_H(Z_s^k) = s_2(d_k) = \pi_1\Big(\prod_{i=k}^{\infty} s_2(d_i)\Big),$$

we see that the case $j = s$ is finished. Now suppose that for $1 < j \leq s$ we have proved the following:

$$\pi_1 \circ \tau_I \circ \partial_H(Z_j^k) = \tau \cdot s_2(d_k), \tag{4}$$

then we prove equation (4) for $j - 1$ (observe that we use here axiom T2):

$$\begin{aligned}
\pi_1 \circ \tau_I \circ \partial_H(Z_{j-1}^k) &= \tau \cdot \pi_1 \circ \tau_I \circ \partial_H(Z_j^k) + s_2(d_k) \\
&= \tau \cdot \big(\tau \cdot s_2(d_k)\big) + s_2(d_k) \\
&= \tau \cdot \pi_1\Big(\prod_{i=k}^{\infty} s_2(d_i)\Big).
\end{aligned}$$

We find thus that equation (4) is valid for all $1 \leq j \leq s$. Now it will be evident from equation (2) that the same holds for $j = 0$; so we see that equation (3) is correct for $n = 1$. Now suppose that (3) is proved up to and including n. We will prove it for $n+1$. Choose $k \geq 1$. Again, let first of all $j = s$. Then we find

$$\begin{aligned}
\pi_{n+1} \circ \tau_I \circ \partial_H(Z_s^k) &= s_2(d_k) \cdot \pi_n \circ \tau_I \circ \partial_H(Z_{s-1}^{k+1}) \\
&\overset{(3)}{=} s_2(d_k) \cdot \tau \cdot \pi_n\Big(\prod_{i=k+1}^{\infty} s_2(d_i)\Big) \\
&= \pi_{n+1}\Big(\prod_{i=k}^{\infty} s_2(d_i)\Big).
\end{aligned}$$

Suppose that for $1 < j \leq s$ we have already proved

$$\pi_{n+1} \circ \tau_I \partial_H(Z_j^k) = \tau \cdot \pi_{n+1}\Big(\prod_{i=k}^{\infty} s_2(d_i)\Big), \tag{5}$$

then we find for $j-1$:

$$\begin{aligned}
\pi_{n+1} \circ \tau_I \partial_H(Z_{j-1}^k) &= \tau \cdot \pi_{n+1} \circ \tau_I \partial_H(Z_j^k) + s_2(d_k) \cdot \pi_n \circ \tau_I \circ \partial_H(Z_{j-2}^{k+1}) \\
&\overset{(3)}{=} \tau \cdot \pi_{n+1} \circ \tau_I \partial_H(Z_j^k) + s_2(d_k) \cdot \tau \pi_n\Big(\prod_{i=k+1}^{\infty} s_2(d_i)\Big) \\
&\overset{(5)}{=} \tau \cdot \pi_{n+1}\Big(\prod_{i=k}^{\infty} s_2(d_i)\Big).
\end{aligned}$$

So for $1 \leq j \leq s$ we find that equation (5) holds. Now it is trivial to see with the aid of equation (2) that equation (5) is valid for all $0 \leq j \leq s$. But this means that equation (3) is correct for $n+1$. So we find that this equation holds for all $n \geq 1$. Take, in particular, $k = 1$ and $j = 0$. Then we find for all $n \geq 1$:

$$\pi_n \circ \tau_I \circ \partial_H\Big(B_s \parallel \prod_{i=1}^{\infty} s_1(d_i)\Big) = \tau \cdot \pi_n\Big(\prod_{i=1}^{\infty} s_2(d_i)\Big);$$

thus, using *AIP*, we find that theorem 4.4.2 is proved since $(d_i)_i \in D^\omega$ was arbitrary chosen.

Remarks (4.4.3)

First of all, we see that we did not use in the proof of theorem (4.4.2) any of the theory that we developed so far. The main reason for giving this proof is that it illustrates the problem that will occur if the capacity of the buffer tends to infinity: we see that while we are busy with the induction on n, we need a "small" reversed induction on j; if the capacity of the buffer becomes infinite—we have a queue with unbounded capacity—we can no longer make this reversed induction on j. Therefore, we will need another method to solve this problem.

Secondly, we also see that the above proof makes heavily use of axiom T2. If we have branching bisimulation (q.v. [**23**]), we can not just copy the above proof since in there we do not have T2. In that case we can use methods that will follow hereinafter (when we will show that a queue with unbounded capacity satisfies our correctness criterion) and then we can imitate the proof of (4.4.4) to obtain the result of theorem (4.4.2). This will be done in theorem (4.4.7).

We will give a guarded recursive specification of a FIFO queue Q over a data set D with input channel 1 and output channel 2 (see figure 4.5). We suppose that the data set contains more than one element. It is known that an infinity guarded recursive specification of Q can be given with the subsequent equations:

$$Q = Q_\lambda = \sum_{d \in D} r_1(d) \cdot Q_d, \tag{6}$$

$$Q_{d*\sigma} = s_2(d) \cdot Q_\sigma + \sum_{e \in D} r_1(e) \cdot Q_{d*\sigma*e}, \tag{7}$$

for any word $\sigma \in D^*$ and any $d \in D$. Here, we use $\lambda \in D^*$ for the empty word. The asterisk $(*)$ in equation (7) stands for the concatenation of words.

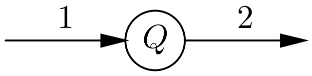

Figure 4.5. A queue Q with input channel 1 and output channel 2.

Below, we will prove that the queue is a correct communication protocol. But first, we will provide in figure 4.6 a picture of the left-hand side of the following equation (8), in which we can see that all the "lines" that go up in the graph are labeled with a τ.

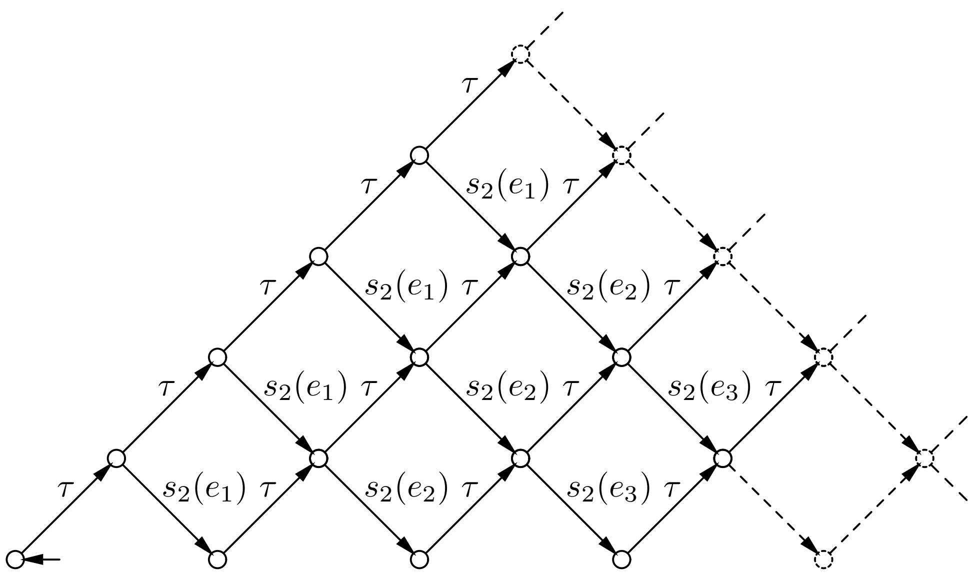

Figure 4.6. A graph of the left-hand side of equation (8)

Theorem (4.4.4)

The queue Q is a correct communication protocol.

Proof. Let $(e_i)_i \in D^\omega$. Let $H = \{r_1(d), s_1(d) : d \in D\}$ and $I = \{c_1(d) : d \in D\}$. In accord with definition (4.4.1), we are to show that

$$\tau_I \circ \partial_H\Big(Q_\lambda \parallel \prod_{i=1}^{\infty} s_1(e_i)\Big) = \tau \cdot \prod_{i=1}^{\infty} s_2(e_i). \tag{8}$$

We will show this with the use of *GIP*. So let $J = \{s_2(d) : d \in D\}$. We will show that for all $n \geq 1$, for all $(e_i)_i \in D^\omega$ and for all $k \geq 1$ the following holds:

$$\pi_J^n \circ \tau_I \circ \partial_H\Big(Q_\lambda \parallel \prod_{i=1}^{\infty} s_1(e_i)\Big) = \tau \cdot \prod_{i=1}^{n} s_2(e_i), \tag{9}$$

$$\pi_J^n \circ \tau_I \circ \partial_H\Big(Q_{e_1 * \cdots * e_k} \parallel \prod_{i=k+1}^{\infty} s_1(e_i)\Big) = \tau \cdot \prod_{i=1}^{n} s_2(e_i). \tag{10}$$

We will prove equations (9) and (10) with induction on n. But first we will do some preliminary calculations. Therefore we will introduce the following abbreviations (for all $k \geq 1$):

$$Z_0 = Q_\lambda \parallel \prod_{i=1}^{\infty} s_1(e_i),$$

$$Z_k = Q_{e_1 * \cdots * e_k} \parallel \prod_{i=k+1}^{\infty} s_1(e_i).$$

Now we will calculate for all $k \geq 0 : \partial_H(Z_k)$. First we handle the case $k = 0$:

$$\begin{aligned}
\partial_H(Z_0) &= \partial_H\Big(Q_\lambda \parallel \prod_{i=1}^{\infty} s_1(e_i)\Big)\\
&= \partial_H\Big(Q_\lambda \mid \prod_{i=1}^{\infty} s_1(e_i)\Big)\\
&= c_1(e_1) \cdot \partial_H\Big(Q_{e_1} \parallel \prod_{i=2}^{\infty} s_1(e_i)\Big)\\
&= c_1(e_1) \cdot \partial_H(Z_1).
\end{aligned}$$

Now we treat the case $k \geq 1$:

$$\partial_H(Z_k) = \partial_H\Big(Q_{e_1 * \cdots * e_k} \parallel \prod_{i=k+1}^{\infty} s_1(e_i)\Big)$$

$$= \partial_H\Big(Q_{e_1*\cdots*e_k} \mathbin{⌊\!⌊} \prod_{i=k+1}^{\infty} s_1(e_i)\Big)$$
$$+ \partial_H\Big(Q_{e_1*\cdots*e_k} \mid \prod_{i=k+1}^{\infty} s_1(e_i)\Big)$$
$$= s_2(e_1) \cdot \partial_H\Big(Q_{e_2*\cdots*e_k} \parallel \prod_{i=k+1}^{\infty} s_1(e_i)\Big)$$
$$+ c_1(e_{k+1}) \cdot \partial_H\Big(Q_{e_1*\cdots*e_{k+1}} \parallel \prod_{i=k+2}^{\infty} s_1(e_i)\Big)$$
$$= s_2(e_1) \cdot \partial_H\Big(Q_{e_2*\cdots*e_k} \parallel \prod_{i=k+1}^{\infty} s_1(e_i)\Big) + c_1(e_{k+1}) \cdot \partial_H(Z_{k+1})$$

Now we will define an auxiliary linear unary operator as follows: Let $\{b, t\}$ be a set of function names and fix an arbitrary $i \in I$. Consider the linear functional specification hereinafter:

$$\begin{aligned} E(b,t) = {} & \big\{\, b(a) = a : a \in A \setminus I \,\big\} \\ & \cup \big\{\, b(a) = i : a \in I \,\big\} \\ & \cup \big\{\, t(a) = a : a \in A \setminus I \,\big\} \\ & \cup \big\{\, t(a) = \tau : a \in I \,\big\} \\ & \cup \big\{\, b(a \cdot x) = b(a) \cdot b(x) : a \in A \setminus J \,\big\} \\ & \cup \big\{\, b(a \cdot x) = b(a) \cdot t(x) : a \in J \,\big\} \\ & \cup \big\{\, t(a \cdot x) = t(a) \cdot t(x) : a \in A \,\big\}. \end{aligned}$$

According to *ODP* there is a valuation $\varphi : \{b, t\} \longrightarrow F$ that solves the system $E(b, t)$. Let us say $\varphi(b) = \beta$ and $\varphi(t) = \gamma$. We can see from the defining equations for γ that this operator behaves just like τ_I. We will show that $\gamma = \tau_I$. Let n be a function name. Consider the following linear functional specification:

$$\begin{aligned} E(n) = {} & \big\{\, n(a) = a : a \in A \setminus I \,\big\} \\ & \cup \big\{\, n(a) = \tau : a \in I \,\big\} \\ & \cup \big\{\, n(a \cdot x) = n(a) \cdot n(x) : a \in A \,\big\}. \end{aligned}$$

It will be clear that τ_I solves this system. But from the defining equations of γ it is also clear that γ solves this system, too. Thus we can conclude with the aid of *OSP* that $\gamma = \tau_I$. We find thus the following functional equation for all $x \in P$ and $a \in J$:

$$\beta(a \cdot x) = \beta(a) \cdot \tau_I(x). \tag{11}$$

In fact we see in this equation the crux of the operator β. We can see this operator as some sort of "booby-trap". Only if the dangerous atomic actions pass, this operator will "detonate" and it becomes the abstraction operator afterwards. In all other circumstances only the abstraction set will be reduced by replacing all the internal actions by a fixed $i \in I$. This technique is called pre-abstraction and is studied in [**4**]. In the body of the proof it will be fine to have this situation but in the end we will treat all the atomic actions equally. So we will have to defuse this booby-trap. We will show hereinafter that $\tau_I \circ \beta = \tau_I$. Let thereto $\{r, s\}$ be a set of function names. Consider the following linear functional specification:

$$
\begin{aligned}
E(r,s) = &\left\{ r(a) = s(a) = a : a \in A \setminus I \right\} \\
&\cup \left\{ r(a) = s(a) = \tau : a \in I \right\} \\
&\cup \left\{ r(a \cdot x) = r(a) \cdot r(x) : a \in A \setminus J \right\} \\
&\cup \left\{ r(a \cdot x) = r(a) \cdot s(x) : a \in J \right\} \\
&\cup \left\{ s(a \cdot x) = s(a) \cdot s(x) : a \in A \right\}.
\end{aligned}
$$

We will show that $(\tau_I \circ \beta, \tau_I)$ is a solution for this system. First we will handle the boundary conditions. We see that

$$\tau_I \circ \beta(a) = \tau_I(a),$$

since $\beta(a) = a$ for all $a \in A \setminus I$ and $\beta(a) = i$ for all $a \in I$. Thus the boundary conditions are satisfied. Now we will treat the functional equations. It is evident that the functional equations for s are satisfied for all atomic actions. So we will only show that the functional equations for r are valid. Let $a \notin J$. Then we find:

$$
\begin{aligned}
\tau_I \circ \beta(a \cdot x) &= \tau_I\big(\beta(a) \cdot \beta(x)\big) \\
&= \tau_I \circ \beta(a) \cdot \tau_I \circ \beta(x).
\end{aligned}
$$

Now let $a \in J$. Then we obtain with the aid of (11):

$$
\begin{aligned}
\tau_I \circ \beta(a \cdot x) &= \tau_I\big(\beta(a) \cdot \tau_I(x)\big) \\
&= \tau_I \circ \beta(a) \cdot \tau_I \circ \tau_I(x) \\
&= \tau_I \circ \beta(a) \cdot \tau_I(x).
\end{aligned}
$$

We used here the following fact: $\tau_I \circ \tau_I = \tau_I$. It is trivial to see that $\tau_I \circ \tau_I$ is also a solution for the linear functional specification $E(n)$ that we defined hereinbefore, so with the aid of *OSP* we find that $\tau_I \circ \tau_I = \tau_I$. We find thus that the functional equations are valid. If we take on the other hand $r = s = \tau_I$ then it is not difficult to see that this solves the system $E(r, s)$, too. So in accordance with *OSP* we may conclude that these solutions must be equal. In particular we find:

$$\tau_I \circ \beta = \tau_I. \tag{12}$$

At this point we will return to the calculations that we were busy with. Our aim is to calculate $\tau_I \circ \partial_H(Z_k)$. If we look at the equations deduced for $\partial_H(Z_k)$ so far, we can see that if we apply the abstraction operator to them, we obtain unguarded recursive equations. Hence, this type of calculating is in a certain way too coarse. Therefore, we have introduced β. We can apply it to the equations and we still have guardedness since for all atomic actions $a \in A$ we have $\beta(a) \in A$. In certain subterms where we actually want an abstraction operator, this β will transform into τ_I after passing the "right" atomic actions.

Now we will calculate for all $k \geq 0 : \beta \circ \partial_H(Z_k)$. Again we will make a case distinction. First, let $k = 0$. Then we find at once

$$\beta \circ \partial_H(Z_0) = i \cdot \beta \circ \partial_H(Z_1).$$

We will handle the case $k \geq 1$. We will use (11) here.

$$\begin{aligned}\beta \circ \partial_H(Z_k) = s_2(e_1) \cdot \tau_I \circ \partial_H\Big(Q_{e_2 * \cdots * e_k} \parallel \prod_{i=k+1}^{\infty} s_1(e_i)\Big) \\ + i \cdot \beta \circ \partial_H(Z_{k+1}).\end{aligned}$$

At this point we will calculate for all $n \geq 1$ and for all $k \geq 0$ the nth projection of $\beta \circ \partial_H(Z_k)$. First, we will consider the case $k = 0$.

$$\pi_J^n \circ \beta \circ \partial_H(Z_0) = i \cdot \pi_J^n \circ \beta \circ \partial_H(Z_1)$$

Now we will treat the case $k \geq 1$. Here we will have two subcases: $n = 1$ and $n > 1$. First, let $n = 1$ and consider the following

$$\pi_J^1 \circ \beta \circ \partial_H(Z_k) = s_2(e_1) + i \cdot \pi_J^1 \circ \beta \circ \partial_H(Z_{k+1}). \tag{13}$$

Now let $n \geq 1$. Then we calculate

$$\begin{aligned}\pi_J^{n+1} \circ \beta \circ \partial_H(Z_k) = s_2(e_1) \cdot \pi_J^n \circ \tau_I \circ \partial_H\Big(Q_{e_2 * \cdots * e_k} \parallel \prod_{i=k+1}^{\infty} s_1(e_i)\Big) \\ + i \cdot \pi_J^{n+1} \circ \beta \circ \partial_H(Z_{k+1}).\end{aligned} \tag{14}$$

Now we have all the prerequisites needed to establish the induction mentioned in the beginning of the proof. Recall that we proposed to verify that equations (9) and (10) hold for all $n \geq 1$. We will verify the basis of our induction. Let $n = 1$. Consider the following guarded recursive specification E_1:

$$E_1 = \big\{\, X_k = s_2(e_1) + i \cdot X_{k+1} \mid k \geq 1 \,\big\}.$$

With the aid of equation (13) we immediately see that, if we put for all $k \geq 1$:

$$X_k = \pi_J^1 \circ \beta \circ \partial_H(Z_k),$$

this solves the guarded recursive specification E_1. Now consider the guarded recursive specification F_1:

$$F_1 = \{\, X = s_2(e_1) + i \cdot X \,\}.$$

According to *RDP* there is a solution for this system. Let us say that x is a solution for F_1. If we put $X_k = x$ for all $k \geq 1$, we see that x is also a solution for the guarded recursive specification E_1. In accordance with *RSP* we may conclude now that for all $k \geq 1$:

$$x = \pi_J^1 \circ \beta \circ \partial_H(Z_k).$$

Since x is the solution for F_1, we see that the following equation is valid:

$$x = s_2(e_1) + i \cdot x.$$

The guard i is an internal atomic action. This means that we may apply *KFAR* on this equation. This yields: $\tau_I(x) = \tau \cdot s_2(e_1)$. So we find

$$\begin{aligned}
\tau \cdot s_2(e_1) &= \tau_I(x) \\
&= \tau_I \circ \pi_J^1 \circ \beta \circ \partial_H(Z_k) \\
&= \pi_J^1 \circ \tau_I \circ \beta \circ \partial_H(Z_k) && \text{because of (4.3.12)} \\
&= \pi_J^1 \circ \tau_I \circ \partial_H(Z_k). && \text{because of (12)}
\end{aligned}$$

Thus, we see that equation (10) holds for $n = 1$. Now we will show that equation (9) is correct for $n = 1$. Let thereto $k = 0$ and consider the subsequent display.

$$\begin{aligned}
\tau_I \circ \pi_J^1 \circ \beta \circ \partial_H(Z_0) &= \tau \cdot \tau_I \circ \pi_J^1 \circ \beta \circ \partial_H(Z_1) \\
&= \tau \cdot \pi_J^1 \circ \tau_I \circ \partial_H(Z_1) \\
&= \tau \cdot \tau \cdot s_2(e_1) \\
&= \tau \cdot s_2(e_1),
\end{aligned}$$

since we just have seen above that for all $k \geq 1 : \pi_J^1 \circ \tau_I \circ \partial_H(Z_k) = \tau \cdot s_2(e_1)$. Thus, we see that both equations (9) and (10) are proved for $n = 1$. Now let us assume that these equations are valid for all $(e_i)_i \in D^\omega$ and $k \geq 1$ for a certain $n \geq 1$. Then we will prove these equations for $n + 1$. This proof will be more or less the same as the verification of the case $n = 1$. We will work out equation (14) a bit more. Let for all $k \geq 1 : e'_i = e_{i+1}$ (we will use the induction hypothesis for $(e'_i)_i \in D^\omega$ and $k-1 \geq 0$). Then we find

$$\begin{aligned}\pi_J^{n+1} \circ \beta \circ \partial_H(Z_k) &= s_2(e_1) \cdot \pi_J^n \circ \tau_I \circ \partial_H\Big(Q_{e'_1 * \cdots * e'_{k-1}} \parallel \prod_{i=k}^{\infty} s_1(e'_i)\Big) \\ &\quad + i \cdot \pi_J^{n+1} \circ \beta \circ \partial_H(Z_{k+1}) \\ &= s_2(e_1) \cdot \tau \cdot \prod_{i=1}^{n} s_2(e'_i) + i \cdot \pi_J^{n+1} \circ \beta \circ \partial_H(Z_{k+1}) \\ &= \prod_{i=1}^{n+1} s_2(e_i) + i \cdot \pi_J^{n+1} \circ \beta \circ \partial_H(Z_{k+1}). \qquad (15)\end{aligned}$$

Now consider the following guarded recursive specification E_2:

$$E_2 = \Big\{ Y_k = \prod_{i=1}^{n+1} s_2(e_i) + i \cdot Y_{k+1} \;\Big|\; k \geq 1 \Big\}.$$

We see with the aid of equation (15), if we put for all $k \geq 1$

$$Y_k = \pi_J^{n+1} \circ \beta \circ \partial_H(Z_k)$$

that this is a solution for the guarded recursive specification E_2. Now consider the guarded recursive specification F_2:

$$F_2 = \Big\{ Y = \prod_{i=1}^{n+1} s_2(e_i) + i \cdot Y \Big\}.$$

With the aid of *RDP* we know that this system has a solution. Let us say that y is the solution for F_2. If we put for all $k \geq 1 : Y_k = y$, then we see that this also solves the guarded recursive specification E_2. So with the aid of *RSP* we conclude that for all $k \geq 1$:

$$y = \pi_J^{n+1} \circ \beta \circ \partial_H(Z_k).$$

Observe that the following equation holds

$$y = i \cdot y + \prod_{i=1}^{n+1} s_2(e_i),$$

so we may apply *KFAR* since $i \in I$. We find thus that

$$\tau_I(y) = \tau \cdot \prod_{i=1}^{n+1} s_2(e_i).$$

On the other hand, we find for all $k \geq 1$

$$\begin{aligned}
\tau_I(y) &= \tau_I \circ \pi_J^{n+1} \circ \beta \circ \partial_H(Z_k) \\
&= \pi_J^{n+1} \circ \tau_I \circ \beta \circ \partial_H(Z_k) && \text{because of (4.3.12)} \\
&= \pi_J^{n+1} \circ \tau_I \circ \partial_H(Z_k). && \text{because of (12)}
\end{aligned}$$

Combining these two inferences, we see that equation (10) is valid for $n+1$. Now we will handle the case $k = 0$. Since

$$\pi_J^{n+1} \circ \tau_I \circ \partial_H(Z_0) = \tau \cdot \pi_J^{n+1} \circ \tau_I \circ \partial_H(Z_1),$$

we find, using the just derived results for $k = 1$, that

$$\pi_J^{n+1} \circ \tau_I \circ \partial_H(Z_0) = \tau \cdot \tau \cdot \prod_{i=1}^{n+1} s_2(e_i) = \tau \cdot \prod_{i=1}^{n+1} s_2(e_i).$$

This means that equation (9) is also valid. This will end proof of equations (9) and (10). Now we find in particular for all $n \geq 1$ and $k = 0$ that equation (9) is valid. If we adjust this equation a little as is done below

$$\pi_J^n \circ \tau_I \circ \partial_H\Big(Q_\lambda \parallel \prod_{i=1}^{\infty} s_1(e_i)\Big) = \pi_J^n\Big(\tau \cdot \prod_{i=1}^{\infty} s_2(e_i)\Big),$$

we may apply *GIP* and we find

$$\tau_I \circ \partial_H\Big(Q_\lambda \parallel \prod_{i=1}^{\infty} s_1(e_i)\Big) = \tau \cdot \prod_{i=1}^{\infty} s_2(e_i).$$

According to definition (4.4.1), we may conclude that Q_λ is a correct communication protocol. This will end the proof of 4.4.4.

Remark (4.4.5)

As we can see in the proof of the former theorem, we did not use axiom T2. In fact, we only used axiom T1, which means that this proof can also be given if we had used branching τ-laws.

We will formulate a lemma on β for later reference.

Lemma (4.4.6) *The Selective Abstraction Lemma*

Let $S, I \subseteq A$. Then there is a linear unary operator $\beta(I,S) \in F$ with the following properties:

(i) $\beta(I,S)(a) = a \qquad (\forall a \in A)$

(ii) $\beta(I,S)(a \cdot x) = a \cdot \beta(I,S)(x) \qquad (\forall a \in A \setminus S, \forall x \in P)$

(iii) $\beta(I,S)(a \cdot x) = a \cdot \tau_I(x) \qquad (\forall a \in S, \forall x \in P)$

(iv) $\tau_I \circ \beta(I,S) = \tau_I$

Proof. Replace in the proof of theorem (4.4.4) the set J for the set S and do not rename internal actions to a specific one. Then the introduction of the linear unary operator β will satisfy the requirements of 4.4.6. And we can take $\beta(I,S) = \beta$. This will end the proof of our lemma.

We will supply another proof of theorem (4.4.2) in which we will not use axiom T2. We will use the method that we established in the proof of theorem (4.4.4).

Theorem (4.4.7)

Let B_s be the s-bit buffer of page 172. For all $(d_i)_i \in D^\omega$ the following holds:

$$\tau_I \circ \partial_H\Big(B_s \parallel \prod_{i=1}^{\infty} s_1(d_i)\Big) = \tau \cdot \prod_{i=1}^{\infty} s_2(d_i).$$

Proof. We will (tacitly) use the notations as they appear in the former theorems. It is very easy to see that the following equations are valid $(1 \le j < s)$.

$$\begin{aligned}
\partial_H(Z_0^k) &= c_1(d_k) \cdot \partial_H(Z_1^k)\\
\partial_H(Z_j^k) &= c_1(d_{k+j}) \cdot \partial_H(Z_{j+1}^k) + s_2(d_k) \cdot \partial_H(Z_{j-1}^{k+1})\\
\partial_H(Z_s^k) &= s_2(d_k) \cdot \partial_H(Z_{s-1}^{k+1}).
\end{aligned}$$

In contrast to the method used in theorem (4.4.2), we will follow here the method of theorem (4.4.4). So let $J = \{s_2(d) : d \in D\}$ and $\beta = \beta(I,J)$; see lemma (4.4.6). Thus, we find using equation (11)

$$\begin{aligned}
\beta \circ \partial_H(Z_0^k) &= c_1(d_k) \cdot \beta \circ \partial_H(Z_1^k)\\
\beta \circ \partial_H(Z_j^k) &= c_1(d_{k+j}) \cdot \beta \circ \partial_H(Z_{j+1}^k) + s_2(d_k) \cdot \tau_I \circ \partial_H(Z_{j-1}^{k+1})\\
\beta \circ \partial_H(Z_s^k) &= s_2(d_k) \cdot \tau_I \circ \partial_H(Z_{s-1}^{k+1}).
\end{aligned}$$

We will prove the following claim with induction on n.

$$\forall k, n \ge 1, \forall 0 \le j \le s : \tag{16}$$

$$\pi_J^n \circ \tau_I \circ \partial_H(Z_j^k) = \begin{cases} \tau \cdot \pi_J^n\Big(\prod_{i=k}^{\infty} s_2(d_i)\Big), & \text{if } 0 \le j < s;\\ \pi_J^n\Big(\prod_{i=k}^{\infty} s_2(d_i)\Big), & \text{if } j = s.\end{cases}$$

First, we will prove equation (16) for $n = 1$. For $j = s$ we immediately see that this is the case. Now assume that $1 \leq j < s$. We will calculate

$$\pi_J^1 \circ \beta \circ \partial_H(Z_j^k) = c_1(d_{k+j}) \cdot \pi_J^1 \circ \beta \circ \partial_H(Z_{j+1}^k) + s_2(d_k).$$

Now consider the following guarded recursive specification E_1.

$$E_1 = \left\{\, X_j = c_1(d_{k+j}) \cdot X_{j+1} + s_2(d_k),\ X_s = s_2(d_k) \mid 1 \leq j < s \,\right\}$$

It will be clear that if we put $X_j = \pi_J^1 \circ \beta \circ \partial_H(Z_j^k)$ for all $1 \leq j \leq s$, this will solve the system E_1. Now contemplate the guarded recursive specification F_1:

$$F_1 = \left\{\, X = c_1(d_{k+j}) \cdot X + s_2(d_k) \,\right\}$$

and let x be a solution for this system accordingly to *RDP*. If we put for all $1 \leq j < s : X_j = x$, we see that this will solve the system E_1, too. So with the help of *RSP* we may conclude for all $1 \leq j < s$:

$$x = \pi_J^1 \circ \beta \circ \partial_H(Z_j^k).$$

Now we will apply *KFAR* on the equation $x = c_1(d_{k+j}) \cdot x + s_2(d_k)$ and we find thus $\tau_I(x) = \tau \cdot s_2(d_k)$. It is easy to see that, using equation (12), the following holds:

$$\begin{aligned}
\tau \cdot s_2(d_k) &= \tau_I(x) \\
&= \tau_I \circ \pi_J^1 \circ \beta \circ \partial_H(Z_j^k) \\
&= \pi_J^1 \circ \tau_I \circ \beta \circ \partial_H(Z_j^k) \\
&= \pi_J^1 \circ \tau_I \circ \partial_H(Z_j^k).
\end{aligned}$$

So we find that the claim holds for $1 \leq j \leq s$. It is trivial to see that it is also valid for $j = 0$. Thus we see that equation (16) is correct for $n = 1$. Suppose that this claim holds for a certain $n \geq 1$ then we prove it for $n + 1$. First we take $j = s$:

$$\begin{aligned}
\pi_J^{n+1} \circ \tau_I \circ \partial_H(Z_s^k) &= s_2(d_k) \cdot \pi_J^n \circ \tau_I \circ \partial_H(Z_{s-1}^{k+1}) \\
&\overset{(16)}{=} s_2(d_k) \cdot \tau \cdot \pi_J^n\Big(\prod_{i=k+1}^{\infty} s_2(d_i)\Big) \\
&= \pi_J^{n+1}\Big(\prod_{i=k}^{\infty} s_2(d_i)\Big).
\end{aligned}$$

Now we take $1 \leq j < s$. Consider the following guarded recursive specification E_2:

$$E_2 = \left\{\, Y_j = c_1(d_{k+j}) \cdot Y_{j+1} + \prod_{i=k}^{k+n+1} s_2(d_i), Y_s = \prod_{i=k}^{k+n+1} s_2(d_i) \mid 1 \leq j < s \,\right\}.$$

We see at once that, if we put $Y_j = \pi_J^{n+1} \circ \beta \circ \partial_H(Z_j^k)$ for all $1 \leq j < s$, this solves the system E_2 using the induction hypothesis. Now consider also the guarded recursive specification F_2

$$F_2 = \Big\{ Y = c_1(d_{k+j}) \cdot Y + \prod_{i=k}^{k+n+1} s_2(d_i) \Big\}.$$

With the aid of *RDP* we know that this guarded recursive specification has a solution, say y. If we put for all $1 \leq j < s : Y_j = y$ then we see that this will solve the system E_2. But then we find with *RSP* that

$$y = \pi_J^{n+1} \circ \beta \circ \partial_H(Z_j^k).$$

If we apply *KFAR* on the equation

$$y = c_1(d_{k+j}) \cdot y + \prod_{i=k}^{k+n+1} s_2(d_i),$$

we find that

$$\tau_I(y) = \tau \cdot \prod_{i=k}^{k+n+1} s_2(d_i).$$

So we can deduce:

$$\begin{aligned} \tau \cdot \prod_{i=k}^{k+n+1} s_2(d_i) &= \tau_I \circ \pi_J^{n+1} \circ \beta \circ \partial_H(Z_j^k) \\ &= \pi_J^{n+1} \circ \tau_I \circ \beta \circ \partial_H(Z_j^k) \\ &= \pi_J^{n+1} \circ \tau_I \circ \partial_H(Z_j^k). \end{aligned}$$

So we see that equation (16) is correct for $1 \leq j \leq s$. Now it is trivial to see that the same holds for $j = 0$. This concludes the induction step and we find, in particular, for $k = 1$ and $j = 0$ that for all $n \geq 1$:

$$\pi_J^n \circ \tau_I \circ \partial_H\Big(B_s \parallel \prod_{i=1}^{\infty} s_1(d_i)\Big) = \tau \cdot \pi_J^n\Big(\prod_{i=1}^{\infty} s_2(d_i)\Big).$$

Thus, using *GIP*, we finish the proof of 4.4.7.

Next, we will give an example of an alternating bit protocol with a time out. See figure 4.7 hereinafter.

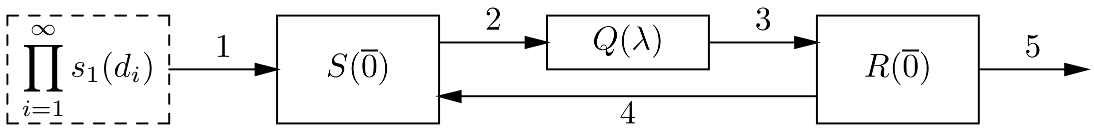

Figure 4.7. An alternating bit protocol with a time out.

The dashed box in figure 4.7 is not part of the protocol. It is a test row. Below, we will give the specification of the components that are present in figure 4.7. But first, we will explain some of the symbols used in the specification. The symbol $\overline{k}$ is common in group theory. It stands for

$$\overline{k} \equiv k \bmod 2 \in \mathbb{Z}/2\mathbb{Z}.$$

It is, in fact, the alternating bit. We use this notation since it will shorten notations somewhat. Furthermore, D is a finite set of datum elements. The abbreviation RA stands for read acknowledgement. The atomic action t stands for *time out* and it is this particular action that justifies the use of a queue with an unbounded capacity. The set E is the following:

$$E = \{d\overline{k} : d \in D, \overline{k} \in \mathbb{Z}/2\mathbb{Z}\}.$$

The abbreviation SM means send message and SA send acknowledgement.

$$\begin{aligned}
S(\overline{k}) &= \sum_{d \in D} r_1(d) \cdot S(d\overline{k}) \\
S(d\overline{k}) &= s_2(d\overline{k}) \cdot RA(d\overline{k}) \\
RA(d\overline{k}) &= r_4(\overline{k}) \cdot S(\overline{k+1}) + \big(r_4(\overline{k+1}) + t\big) \cdot S(d\overline{k}) \\
Q(\lambda) &= \sum_{x \in E} r_2(x) \cdot Q(x) \\
Q(x * \sigma) &= \sum_{y \in E} r_2(y) \cdot Q(x * \sigma * y) + s_3(x) \cdot Q(\sigma) \\
R(\overline{k}) &= \sum_{d \in D} r_3(d\overline{k}) \cdot SM(d\overline{k}) + \sum_{d \in D} r_3\big(d(\overline{k+1})\big) \cdot SA(\overline{k+1}) \\
SM(d\overline{k}) &= s_5(d) \cdot SA(\overline{k}) \\
SA(\overline{k}) &= s_4(\overline{k}) \cdot R(\overline{k+1})
\end{aligned}$$

Below, we will enumerate the encapsulation set H and the abstraction set I.

$$\begin{aligned}
H = &\left\{\, r_1(d), s_1(d) : d \in D\right\} \\
&\cup \left\{\, r_2(x), r_3(x), s_2(x), s_3(x) : x \in E\right\} \\
&\cup \left\{\, r_4(\overline{k}), s_4(\overline{k}) : \overline{k} \in \mathbb{Z}/2\mathbb{Z} \,\right\} \\
I = &\left\{\, c_1(d) : d \in D \,\right\} \\
&\cup \left\{\, c_2(x), c_3(x) : x \in E \,\right\} \\
&\cup \left\{\, c_4(\overline{k}) : \overline{k} \in \mathbb{Z}/2\mathbb{Z} \,\right\} \cup \{t\}.
\end{aligned}$$

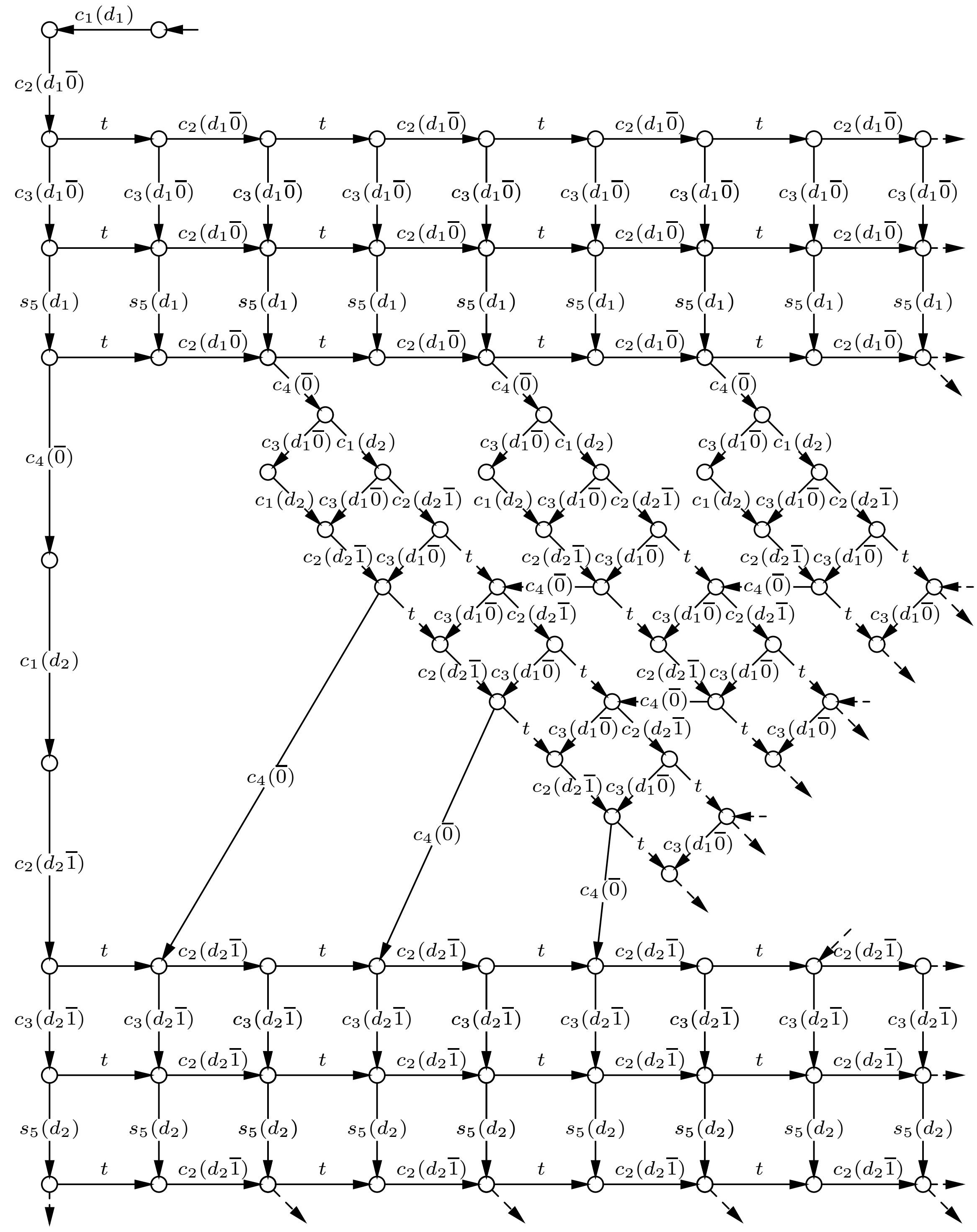

Figure 4.8. A beginning of a graph of $\partial_H\Big(S(\overline{0}) \parallel Q(\lambda) \parallel R(\overline{0}) \parallel \prod_{i=1}^{\infty} s_1(d_i)\Big)$.

Theorem (4.4.8)

$$\tau_I \circ \partial_H\Big(S(\overline{0}) \parallel Q(\lambda) \parallel R(\overline{0}) \parallel \prod_{i=1}^{\infty} s_1(d_i)\Big) = \tau \cdot \prod_{i=1}^{\infty} s_5(d_i).$$

Proof. We will first calculate

$$\partial_H\Big(S(\overline{0}) \parallel Q(\lambda) \parallel R(\overline{0}) \parallel \prod_{i=1}^{\infty} s_1(d_i)\Big).$$

Figure 4.8 (see page 187) is a picture of this process. We will introduce some notations for convenience sake. We will abbreviate $\tilde{d}_{k+1} = d_{k+1}\overline{k}$ and for all $n \geq 1$:

$$\tilde{d}_{k+1}^n = \underbrace{\tilde{d}_{k+1} * \cdots * \tilde{d}_{k+1}}_{n \text{ times}}.$$

If $n = 0$ we will use the convention $x^0 = \lambda$ for all words $x \in E^*$. Let for all $k \geq 0$ and any word $\sigma \in E^*$ the following be given:

$$A_k(\sigma) = S(\overline{k}) \parallel Q(\sigma) \parallel R(\overline{k}) \parallel \prod_{i=k+1}^{\infty} s_1(d_i)$$

$$B_k(\sigma) = S(\tilde{d}_{k+1}) \parallel Q(\sigma) \parallel R(\overline{k}) \parallel \prod_{i=k+2}^{\infty} s_1(d_i)$$

$$C_k(\sigma) = RA(\tilde{d}_{k+1}) \parallel Q(\sigma) \parallel R(\overline{k}) \parallel \prod_{i=k+2}^{\infty} s_1(d_i)$$

$$D_k(\sigma) = RA(\tilde{d}_{k+1}) \parallel Q(\sigma) \parallel SM(\tilde{d}_{k+1}) \parallel \prod_{i=k+2}^{\infty} s_1(d_i)$$

$$E_k(\sigma) = S(\tilde{d}_{k+1}) \parallel Q(\sigma) \parallel SM(\tilde{d}_{k+1}) \parallel \prod_{i=k+2}^{\infty} s_1(d_i)$$

$$F_k(\sigma) = RA(\tilde{d}_{k+1}) \parallel Q(\sigma) \parallel SA(\overline{k}) \parallel \prod_{i=k+2}^{\infty} s_1(d_i)$$

$$G_k(\sigma) = RA(\tilde{d}_{k+2}) \parallel Q(\sigma) \parallel SA(\overline{k}) \parallel \prod_{i=k+3}^{\infty} s_1(d_i).$$

Observe, that we are to evaluate $\partial_H\big(A_0(\lambda)\big)$. If we do so, we will find that there is much more needed. For, we will have to calculate $\partial_H\big(A_k(\lambda)\big)$ for all $k \geq 0$. If we make this calculation, we will see that we will also need to evaluate for all $k, n \geq 0$

$$\partial_H\big(A_{k+1}(\tilde{d}_{k+1}^n)\big).$$

If we calculate this "one" we will find that this is enough to obtain a guarded recursive specification for $\partial_H\big(A_0(\lambda)\big)$ with the abbreviations that we can find above. First, we will give the result of the calculation. Second, with the aid of the guarded recursive specification, consisting of equations (17)–(27) and figure 4.8 of page 187, we will provide some explanation such that it will be not that hard for the reader to verify the result. For all $k \geq 0$ we have the following guarded recursive specification for $\partial_H\big(A_0(\lambda)\big)$.

$$\begin{aligned}
\partial_H\big(A_k(\lambda)\big) &= c_1(d_{k+1}) \cdot \partial_H\big(B_k(\lambda)\big) && (17)\\
\partial_H\big(A_{k+1}(\tilde{d}^n_{k+1})\big) &= c_1(d_{k+2}) \cdot \partial_H\big(B_{k+1}(\tilde{d}^n_{k+1})\big) && (n \geq 1)\\
&\quad + c_3(\tilde{d}_{k+1}) \cdot c_1(d_{k+2})\\
&\quad \cdot c_2(\tilde{d}_{k+2}) \cdot \partial_H\big(G_k(\tilde{d}^{n-1}_{k+1} * \tilde{d}_{k+2})\big) && (18)
\end{aligned}$$

$$\begin{aligned}
\partial_H\big(B_k(\lambda)\big) &= c_2(\tilde{d}_{k+1}) \cdot \partial_H\big(C_k(\tilde{d}_{k+1})\big) && (19)\\
\partial_H\big(B_k(\tilde{d}^n_{k+1})\big) &= c_2(\tilde{d}_{k+1}) \cdot \partial_H\big(C_k(\tilde{d}^{n+1}_{k+1})\big) && (n \geq 1)\\
&\quad + c_3(\tilde{d}_{k+1}) \cdot \partial_H\big(E_k(\tilde{d}^{n-1}_{k+1})\big) && (20)\\
\partial_H\big(B_{k+1}(\tilde{d}^n_{k+1} * \tilde{d}^m_{k+2})\big) &= c_2(\tilde{d}_{k+2}) \cdot \partial_H\big(C_{k+1}(\tilde{d}^n_{k+1} * \tilde{d}^{m+1}_{k+2})\big)\\
&\quad + c_3(\tilde{d}_{k+1}) \cdot c_2(\tilde{d}_{k+2}) && (n \geq 1, m \geq 0)\\
&\quad \cdot \partial_H\big(G_k(\tilde{d}^{n-1}_{k+1} * \tilde{d}^{m+1}_{k+2})\big) && (21)
\end{aligned}$$

$$\begin{aligned}
\partial_H\big(C_k(\tilde{d}^n_{k+1})\big) &= t \cdot \partial_H\big(B_k(\tilde{d}^n_{k+1})\big) && (n \geq 1)\\
&\quad + c_3(\tilde{d}_{k+1}) \cdot \partial_H\big(D_k(\tilde{d}^{n-1}_{k+1})\big) && (22)\\
\partial_H\big(C_{k+1}(\tilde{d}^n_{k+1} * \tilde{d}^m_{k+2})\big) &= t \cdot \partial_H\big(B_{k+1}(\tilde{d}^n_{k+1} * \tilde{d}^m_{k+2})\big) && (n, m \geq 1)\\
&\quad + c_3(\tilde{d}_{k+1}) \cdot \partial_H\big(G_k(\tilde{d}^{n-1}_{k+1} * \tilde{d}^m_{k+2})\big) && (23)
\end{aligned}$$

$$\begin{aligned}
\partial_H\big(D_k(\tilde{d}^n_{k+1})\big) &= t \cdot \partial_H\big(E_k(\tilde{d}^n_{k+1})\big) && (n \geq 0)\\
&\quad + s_5(d_{k+1}) \cdot \partial_H\big(F_k(\tilde{d}^n_{k+1})\big) && (24)
\end{aligned}$$

$$\begin{aligned}
\partial_H\big(E_k(\tilde{d}^n_{k+1})\big) &= c_2(\tilde{d}_{k+1}) \cdot \partial_H\big(D_k(\tilde{d}^{n+1}_{k+1})\big) && (n \geq 0)\\
&\quad + s_5(d_{k+1}) \cdot c_2(\tilde{d}_{k+1}) \cdot \partial_H\big(F_k(\tilde{d}^{n+1}_{k+1})\big) && (25)
\end{aligned}$$

$$\begin{aligned}
\partial_H\big(F_k(\tilde{d}^n_{k+1})\big) &= t \cdot c_2(\tilde{d}_{k+1}) \cdot \partial_H\big(F_k(\tilde{d}^{n+1}_{k+1})\big) && (n \geq 0)\\
&\quad + c_4(\overline{k}) \cdot \partial_H\big(A_{k+1}(\tilde{d}^n_{k+1})\big) && (26)
\end{aligned}$$

$$\begin{aligned}
\partial_H\big(G_k(\tilde{d}^n_{k+1} * \tilde{d}^m_{k+2})\big) &= t \cdot c_2(\tilde{d}_{k+2}) && (n \geq 0, m \geq 1)\\
&\quad \cdot \partial_H\big(G_k(\tilde{d}^n_{k+1} * \tilde{d}^{m+1}_{k+2})\big)\\
&\quad + c_4(\overline{k}) \cdot \partial_H\big(B_{k+1}(\tilde{d}^n_{k+1} * \tilde{d}^m_{k+2})\big) && (27)
\end{aligned}$$

At this point we will explain this large guarded recursive specification with the aid of the figure. We will begin in the first node. At that place there is only one possible action that can be performed: the first element will be taken from the test row (we will go to the left in the graph). This corresponds to equation (17). Still there is one possible action: the test datum will be loaded in the queue (we will go down one node in the graph). This transition corresponds to equation (19). Now we entered the third node and here we have two possible actions: the queue can communicate with the receiver and the datum just read will be removed from the queue (we go one node down in the graph) or a time out can be given (we go to the right in the graph). We will call the first infinite horizontal trace the single BC-trace since in this phase the queue can only contain one type of data. We will discuss this trace. We can see that after the time out the queue can be loaded (we will go to the right) or the single BC-trace can be escaped by communicating with the receiver (we will go one node down in the graph). This behaviour is reiterated every two nodes in the single BC-trace. The equations that correspond to this trace are (20) and (22). As we said we can leave the single BC-trace with a communication (in the graph all the down going arrows are labeled with this action) and we will enter what we will call a DE-trace. We will discuss this trace. In fact, this trace is the same as the single BC-trace except that the down going arrows are labeled with a send action and then we will enter an F-trace. The equations that correspond with the DE-trace are (24) and (25). This F-trace consist of alternating time outs and loadings of the queue. The equation that accompanies this is (26). In the picture we can see that there are two different ways to escape the F-trace. The first possible arrow down connects the F-trace "directly" to the next single BC-trace. Then the already described pattern will be repeated. We will explain the arrows that go southeastwards from the F-trace. As we can see in equation (26) these A-nodes contain old data: the acknowledgement has been performed but due to time outs there are still old datum elements in the queue. From these A-nodes we can take the next element of the test row (we go to the southeast in the graph) and we will enter a double BC-trace or we can communicate with the receiver (we go southwestwardly) and we will enter a G-trace. This corresponds to equation (18). We will discuss the double BC-traces. Now there are old data in the queue and moreover, new data can be added if we stay in this trace: alternating time outs and loading of new data will be performed (we go southeastwards in the graph). As with the single BC-traces we can escape the double one by communicating with the receiver (we go southwestwards from a node in the double BC-trace). This corresponds to the equations (21) and (23). Now we will discuss the G-traces. In this trace we can execute alternating time outs and loadings of the queue (we go southeastwards in the graph). We can escape a G-trace by the execution of an acknowledgement and we enter a double BC-trace with one old datum element less (we go to the left in the graph) or if we are already at the leftmost G-trace the next single BC-trace will be entered (the three long arrows in the graph that connect the leftmost G-trace with the second single BC-trace). The equation that corresponds to the G-traces is (27).

Now that it is plausible how this large guarded recursive specification can be obtained, we will discuss the structure of the sequel of the proof of this theorem. First we will make an assumption on a *DE*-trace for one fixed $p \geq 1$ (which is valid for $p = 1$). Then we will derive from this assumption a conclusion on an *F*-trace. To obtain this result we will need an intermediary result on the double *BC*-traces and the *G*-traces. This will be first deduced from the assumptions on the *DE*-trace. As soon as we verified all this we will show that the conclusion on the *F*-trace itself is a true statement. With this result we will prove that the same yields for the assumptions that we made. In fact, we set off a chain reaction: the assumptions for $p = 1$ are satisfied. This implies that the conclusions are valid for $p = 1$. But with this we can show that the assumptions for $p = 2$ are satisfied so the conclusion is valid for $p = 2$, et cetera. We will do this with induction on p. Once we have all these facts we will deduce a result on the single *BC*-traces and then a small calculation will finish the proof.

Let $J = \{s_5(d) : d \in D\}$. Let $\beta_1 = \beta(I, J)$; see for such operators and their properties lemma (4.4.6). We will use these properties tacitly throughout this verification. Let $p \geq 1$ be fixed. Suppose that for all $k, n \geq 0$ the following holds

$$\pi_J^p \circ \beta_1 \circ \partial_H\big(D_k(\tilde{d}_{k+1}^n)\big) = t \cdot \pi_J^p \circ \beta_1 \circ \partial_H\big(E_k(\tilde{d}_{k+1}^n)\big) + \prod_{i=1}^{p} s_5(d_{k+i}) \quad (28)$$

$$\pi_J^p \circ \beta_1 \circ \partial_H\big(E_k(\tilde{d}_{k+1}^n)\big) = c_2(\tilde{d}_{k+1}) \cdot \partial_H\big(D_k(\tilde{d}_{k+1}^{n+1})\big) + \prod_{i=1}^{p} s_5(d_{k+i}) \quad (29)$$

Then we will prove that for all $k, n \geq 0$

$$\pi_J^p \circ \tau_I \circ \partial_H\big(F_k(\tilde{d}_{k+1}^n)\big) = \tau \cdot \prod_{i=1}^{p} s_5(d_{k+1+i}) \quad (30)$$

In order to prove this we will need an intermediary result that we will verify first. For all $k, l, m \geq 0$ we have

$$\left.\begin{array}{l} \pi_J^p \circ \tau_I \circ \partial_H\big(B_{k+1}(\tilde{d}_{k+1}^l * \tilde{d}_{k+2}^m)\big) \\ \pi_J^p \circ \tau_I \circ \partial_H\big(G_k(\tilde{d}_{k+1}^l * \tilde{d}_{k+2}^{m+1})\big) \end{array}\right\} = \tau \cdot \prod_{i=1}^{p} s_5(d_{k+1+i}) \quad (31)$$

We will prove (31) with induction on l. So let $l = 0$ and fix an arbitrary $k \geq 0$. Consider the following guarded recursive specification E_1.

$$E_1 = \Big\{\, X_n^1 = t \cdot Y_n^1 + \prod_{i=1}^{p} s_5(d_{k+1+i})$$

$$Y_n^1 = c_2(\tilde{d}_{k+2}) \cdot X_{n+1}^1 + \prod_{i=1}^{p} s_5(d_{k+1+i}) \;\Big|\; n \geq 0 \,\Big\}.$$

Because of the assumptions that we made in equations (28) and (29) we see that if we put for all $n \geq 0$

$$\begin{aligned} X_n^1 &= \pi_J^p \circ \beta_1 \circ \partial_H\big(D_{k+1}(\tilde{d}_{k+2}^n)\big) \\ Y_n^1 &= \pi_J^p \circ \beta_1 \circ \partial_H\big(E_{k+1}(\tilde{d}_{k+2}^n)\big), \end{aligned}$$

this is a solution for the guarded recursive specification E_1. The subsequent ratiocination will frequently occur in this proof. In fact, we already applied this technique in the former theorems. So it will be convenient to give it a name for later reference. We will baptize it "a collapsing-argument". Now consider the following guarded recursive specification F_1, which is "E_1 without n".

$$\begin{aligned} F_1 = \big\{\, X_1 &= t \cdot Y_1 + \prod_{i=1}^{p} s_5(d_{k+1+i}) \\ Y_1 &= c_2(\tilde{d}_{k+2}) \cdot X_1 + \prod_{i=1}^{p} s_5(d_{k+1+i}) \,\big\}. \end{aligned}$$

According to *RDP* there is a solution for F_1, say (x_1, y_1). So we have two equations

$$\begin{aligned} x_1 &= t \cdot y_1 + \prod_{i=1}^{p} s_5(d_{k+1+i}) \\ y_1 &= c_2(\tilde{d}_{k+2}) \cdot x_1 + \prod_{i=1}^{p} s_5(d_{k+1+i}). \end{aligned}$$

To these equations we may apply $KFAR_2$ (see section 4.2 or definition (1.2.7)) and we find

$$\tau_I(x_1) = \tau_I(y_1) = \tau \cdot \prod_{i=1}^{p} s_5(d_{k+1+i}). \tag{32}$$

It will be clear that, if we put for all $n \geq 0$

$$X_n^1 = x_1, \quad Y_n^1 = y_1,$$

this will solve the system E_1, too. So with the aid of *RSP* we find that for all $n \geq 0$

$$\begin{aligned} x_1 &= \pi_J^p \circ \beta_1 \circ \partial_H\big(D_{k+1}(\tilde{d}_{k+2}^n)\big) \\ y_1 &= \pi_J^p \circ \beta_1 \circ \partial_H\big(E_{k+1}(\tilde{d}_{k+2}^n)\big). \end{aligned}$$

And we obtain using lemma (4.4.6), theorem (4.3.12) and equation (32) that

$$\left.\begin{aligned} &\pi_J^p \circ \tau_I \circ \partial_H\big(D_{k+1}(\tilde{d}_{k+2}^n)\big) \\ &\pi_J^p \circ \tau_I \circ \partial_H\big(E_{k+1}(\tilde{d}_{k+2}^n)\big) \end{aligned}\right\} = \tau \cdot \prod_{i=1}^{p} s_5(d_{k+1+i}). \tag{33}$$

We will show this for the first equation.

$$\begin{aligned}\tau \cdot \prod_{i=1}^{p} s_5(d_{k+1+i}) &= \tau_I(x_1)\\ &= \tau_I \circ \pi_J^p \circ \beta_1 \circ \partial_H\big(D_{k+1}(\tilde{d}_{k+2}^n)\big)\\ &= \pi_J^p \circ \tau_I \circ \beta_1 \circ \partial_H\big(D_{k+1}(\tilde{d}_{k+2}^n)\big) \quad \text{use (4.3.12)}\\ &= \pi_J^p \circ \tau_I \circ \partial_H\big(D_{k+1}(\tilde{d}_{k+2}^n)\big) \text{ see lemma (4.4.6)}\end{aligned}$$

Such a calculation can be made for E_{k+1}, too. We find thus that (33) is correct.

Now consider the guarded recursive specification E_2

$$\begin{aligned}E_2 = \Big\{\, X_m^2 &= c_2(\tilde{d}_{k+2}) \cdot Y_{m+1}^2 + c_3(\tilde{d}_{k+2}) \cdot \prod_{i=1}^{p} s_5(d_{k+1+i})\\ Y_{m+1}^2 &= t \cdot X_{m+1}^2 + c_3(\tilde{d}_{k+2}) \cdot \prod_{i=1}^{p} s_5(d_{k+1+i}) \;\Big|\; m \geq 0 \,\Big\}.\end{aligned}$$

Let $\beta_2 = \beta\big(I, \{c_3(\tilde{d}_{k+2})\}\big)$; see lemma (4.4.6). It is easy to see with the aid of equations (20), (22) and (33) that if we put for all $m \geq 0$

$$\begin{aligned}X_m^2 &= \pi_J^p \circ \beta_2 \circ \partial_H\big(B_{k+1}(\tilde{d}_{k+2}^m)\big)\\ Y_{m+1}^2 &= \pi_J^p \circ \beta_2 \circ \partial_H\big(C_{k+1}(\tilde{d}_{k+2}^{m+1})\big),\end{aligned}$$

this will solve the system E_2. First, we will show this for the first equation of E_2.

$$\begin{aligned}\pi_J^p \circ \beta_2 \circ \partial_H\big(B_{k+1}(\tilde{d}_{k+2}^m)\big) &\overset{(20)}{=} c_2(\tilde{d}_{k+2}) \cdot \pi_J^p \circ \beta_2 \circ \partial_H\big(C_{k+1}(\tilde{d}_{k+2}^{m+1})\big)\\ &\qquad + c_3(\tilde{d}_{k+2}) \cdot \pi_J^p \circ \tau_I \circ \partial_H\big(E_{k+1}(\tilde{d}_{k+2}^{m-1})\big)\\ &\overset{(33)}{=} c_2(\tilde{d}_{k+2}) \cdot \pi_J^p \circ \beta_2 \circ \partial_H\big(C_{k+1}(\tilde{d}_{k+2}^{m+1})\big)\\ &\qquad + c_3(\tilde{d}_{k+2}) \cdot \tau \cdot \prod_{i=1}^{p} s_5(d_{k+1+i}).\end{aligned}$$

Now we will treat the second equation.

$$\begin{aligned}Y_{m+1}^2 &= \pi_J^p \circ \beta_2 \circ \partial_H\big(C_{k+1}(\tilde{d}_{k+2}^{m+1})\big)\\ &\overset{(22)}{=} t \cdot \pi_J^p \circ \beta_2 \circ \partial_H\big(B_{k+1}(\tilde{d}_{k+2}^{m+1})\big)\\ &\qquad + c_3(\tilde{d}_{k+2}) \cdot \pi_J^p \circ \tau_I \circ \partial_H\big(D_{k+1}(\tilde{d}_{k+2}^m)\big)\\ &\overset{(33)}{=} t \cdot X_{m+1}^2 + c_3(\tilde{d}_{k+2}) \cdot \tau \cdot \prod_{i=1}^{p} s_5(d_{k+1+i}).\end{aligned}$$

We see that this solves indeed the guarded recursive specification E_2. Now we are going to use a collapsing-argument as with E_1: we construct a guarded recursive specification F_2 without m.

$$F_2 = \{\, X_2 = c_2(\tilde{d}_{k+2}) \cdot Y_2 + c_3(\tilde{d}_{k+2}) \cdot \prod_{i=1}^{p} s_5(d_{k+1+i})$$

$$Y_2 = t \cdot X_2 + c_3(\tilde{d}_{k+2}) \cdot \prod_{i=1}^{p} s_5(d_{k+1+i}) \,\}.$$

With *RDP* we see that there is a solution (x_2, y_2) for F_2 that is also a solution for the system E_2. With *KFAR*$_2$ we find that

$$\tau_I(x_2) = \tau_I(y_2) = \tau \cdot \prod_{i=1}^{p} s_5(d_{k+1+i})$$

and with *RSP* we see that for all $m \geq 0$

$$x_2 = \pi_J^p \circ \beta_2 \circ \partial_H\big(B_{k+1}(\tilde{d}^m_{k+2})\big).$$

Thus we have

$$\pi_J^p \circ \tau_I \circ \partial_H\big(B_{k+1}(\tilde{d}^m_{k+2})\big) = \tau \cdot \prod_{i=1}^{p} s_5(d_{k+1+i}). \tag{34}$$

For, consider the following

$$\begin{aligned}
\tau \cdot \prod_{i=1}^{p} s_5(d_{k+1+i}) &= \tau_I(x_2) \\
&= \tau_I \circ \pi_J^p \circ \beta_2 \circ \partial_H\big(B_{k+1}(\tilde{d}^m_{k+2})\big) \\
&= \pi_J^p \circ \tau_I \circ \beta_2 \circ \partial_H\big(B_{k+1}(\tilde{d}^m_{k+2})\big) \quad \text{use (4.3.12)} \\
&= \pi_J^p \circ \tau_I \circ \partial_H\big(B_{k+1}(\tilde{d}^m_{k+2})\big) \text{ see lemma (4.4.6)}
\end{aligned}$$

This means that the first equation of (31) is correct for $l = 0$ and our fixed $k \geq 0$.

Consider the following guarded recursive specification E_3.

$$E_3 = \{\, X^3_m = t \cdot Y^3_m + c_4(\overline{k}) \cdot \prod_{i=1}^{p} s_5(d_{k+1+i})$$

$$Y^3_m = c_2(\tilde{d}_{k+2}) \cdot X^3_{m+1} \;\big|\; m \geq 1 \,\}.$$

Let $\beta_3 = \beta\big(I, \{c_4(\overline{k})\}\big)$. We will show that, if we put for all $m \geq 1$

$$X^3_m = \pi_J^p \circ \beta_3 \circ \partial_H\big(G_k(\tilde{d}^m_{k+2})\big)$$

this will solve the guarded recursive specification E_3. We will only verify this for the first equation of E_3

$$\begin{aligned}
X^3_m &= \pi^p_J \circ \beta_3 \circ \partial_H\big(G_k(\tilde{d}^m_{k+2})\big) \\
&\overset{(27)}{=} t \cdot c_2(\tilde{d}_{k+2}) \cdot \pi^p_J \circ \beta_3 \circ \partial_H\big(G_k(\tilde{d}^{m+1}_{k+2})\big) \\
&\qquad + c_4(\overline{k}) \cdot \pi^p_J \circ \tau_I \circ \partial_H\big(B_{k+1}(\tilde{d}^m_{k+2})\big) \\
&\overset{(34)}{=} t \cdot Y^3_m + c_4(\overline{k}) \cdot \tau \cdot \prod_{i=1}^{p} s_5(d_{k+1+i}).
\end{aligned}$$

Again, we will use a collapsing-argument. Consider the guarded recursive specification F_3 without m.

$$\begin{aligned}
F_3 = \big\{\, X_3 &= t \cdot Y_3 + c_4(\overline{k}) \cdot \prod_{i=1}^{p} s_5(d_{k+1+i}) \\
Y_3 &= c_2(\tilde{d}_{k+2}) \cdot X_3 \,\big\}
\end{aligned}$$

and let (x_3, y_3) be a solution for it. Then it is also a solution for E_3. After using $KFAR_2$ on F_3 we find with (4.3.12) and (4.4.6) that

$$\pi^p_J \circ \tau_I \circ \partial_H\big(G_k(\tilde{d}^m_{k+2})\big) = \tau \cdot \prod_{i=1}^{p} s_5(d_{k+1+i}).$$

So we see that our intermediate result (31) is correct for $l = 0$ since $k \geq 0$ was arbitrarily chosen. Assume that (31) is proved for $l \geq 0$. We will verify it for $l + 1$. Fix a $k \geq 0$ and consider the following guarded recursive specification E_4.

$$\begin{aligned}
E_4 = \big\{\, X^4_m &= c_2(\tilde{d}_{k+2}) \cdot Y^4_{m+1} + c_3(\tilde{d}_{k+1}) \cdot \prod_{i=1}^{p} s_5(d_{k+1+i}) \\
Y^4_{m+1} &= t \cdot X^4_{m+1} + c_3(\tilde{d}_{k+1}) \cdot \prod_{i=1}^{p} s_5(d_{k+1+i}) \;\big|\; m \geq 0 \,\big\}.
\end{aligned}$$

Let $\beta_4 = \beta\big(I, \{c_3(\tilde{d}_{k+1})\}\big)$. Because of equations (21) and (23) and the induction hypothesis (for l), it is easy to see that if we put for all $m \geq 0$

$$\begin{aligned}
X^4_m &= \pi^p_J \circ \beta_4 \circ \partial_H\big(B_{k+1}(\tilde{d}^{l+1}_{k+1} * \tilde{d}^m_{k+2})\big) \\
Y^4_{m+1} &= \pi^p_J \circ \beta_4 \circ \partial_H\big(C_{k+1}(\tilde{d}^{l+1}_{k+1} * \tilde{d}^{m+1}_{k+2})\big),
\end{aligned}$$

this will solve the guarded recursive specification E_4. And we see that with the aid of a collapsing-argument we find, in particular,

$$\pi^p_J \circ \tau_I \circ \partial_H\big(B_{k+1}(\tilde{d}^{l+1}_{k+1} * \tilde{d}^m_{k+2})\big) = \tau \cdot \prod_{i=1}^{p} s_5(d_{k+1+i}).$$

Thus, the first equation of (31) is correct for $l+1$. Because of equation (27) and the above display we see that if we put for all $m \geq 1$

$$X^3_m = \pi^p_J \circ \beta_3 \circ \partial_H\big(G_k(\tilde{d}^{l+1}_{k+1} * \tilde{d}^m_{k+2})\big)$$

that this will solve the system E_3, too. Hence,

$$\pi^p_J \circ \tau_I \circ \partial_H\big(G_k(\tilde{d}^{l+1}_{k+1} * \tilde{d}^m_{k+2})\big) = \tau \cdot \prod_{i=1}^{p} s_5(d_{k+1+i})$$

and our intermediate result (31) is correct for $l+1$. This will end the induction step and we find that (31) is correct.

We were busy with the proof that equation (30) holds if the assumptions on D_k and E_k hold. With the aid of equation (17) we find using our intermediate result for $l = m = 0$

$$\begin{aligned}\pi^p_J \circ \tau_I \circ \partial_H\big(A_{k+1}(\lambda)\big) &= \tau \cdot \pi^p_J \circ \tau_I \circ \partial_H\big(B_{k+1}(\lambda)\big) \\ &= \tau \cdot \prod_{i=1}^{p} s_5(d_{k+1+i}).\end{aligned}$$

And with equation (18) we find, for all $n \geq 1$, using (31) twice

$$\begin{aligned}\pi^p_J \circ \tau_I \circ \partial_H\big(A_{k+1}(\tilde{d}^n_{k+1})\big) &= \tau \cdot \pi^p_J \circ \tau_I \circ \partial_H\big(B_{k+1}(\tilde{d}^n_{k+1})\big) \\ &\quad + \tau \cdot \pi^p_J \circ \tau_I \circ \partial_H\big(G_k(\tilde{d}^{n-1}_{k+1} * \tilde{d}_{k+2})\big) \\ &= \tau \cdot \prod_{i=1}^{p} s_5(d_{k+1+i}) + \tau \cdot \prod_{i=1}^{p} s_5(d_{k+1+i}) \\ &= \tau \cdot \prod_{i=1}^{p} s_5(d_{k+1+i}).\end{aligned}$$

Summarizing, we have seen that for all $n \geq 0$

$$\pi^p_J \circ \tau_I \circ \partial_H\big(A_{k+1}(\tilde{d}^n_{k+1})\big) = \tau \cdot \prod_{i=1}^{p} s_5(d_{k+1+i}). \tag{35}$$

Now consider the following guarded recursive specification E_5.

$$\begin{aligned}E_5 = \Big\{\, X^5_n &= t \cdot Y^5_n + c_4(\overline{k}) \cdot \prod_{i=1}^{p} s_5(d_{k+1+i}) \\ Y^5_n &= c_2(\tilde{d}_{k+1}) \cdot X^5_{n+1} \;\Big|\; n \geq 0 \,\Big\}.\end{aligned}$$

With the aid of equation (26) and (35), it is evident that if we put for all $n \geq 0$

$$X_n^5 = \pi_J^p \circ \beta_3 \circ \partial_H\big(F_k(\tilde{d}_{k+1}^n)\big)$$

this is a solution for the system E_5. Using a collapsing-argument we will find immediately that

$$\pi_J^p \circ \tau_I \circ \partial_H\big(F_k(\tilde{d}_{k+1}^n)\big) = \tau \cdot \prod_{i=1}^{p} s_5(d_{k+1+i}).$$

This is precisely equation (30). Thus, we have proved that if equations (28) and (29) hold that (30) also holds. Now we are going to prove that equation (30) itself holds. We will do that with induction on p. So let $p = 1$. Observe that equations (28) and (29) are valid for $p = 1$ (q.v. equations (24) and (25)). Thus, we find that equation (30) is valid for $p = 1$. Now suppose that (30) holds for $p \geq 1$. Then it is easy to see with the aid of equation (24) and the induction hypothesis for p that

$$\begin{aligned}
\pi_J^{p+1} \circ \beta_1 \circ \partial_H\big(D_k(\tilde{d}_{k+1}^n)\big) &= t \cdot \pi_J^{p+1} \circ \beta_1 \circ \partial_H\big(E_k(\tilde{d}_{k+1}^n)\big) \\
&\quad + s_5(d_{k+1}) \cdot \prod_{i=1}^{p} s_5(d_{k+1+i}) \\
&= t \cdot \pi_J^{p+1} \circ \beta_1 \circ \partial_H\big(E_k(\tilde{d}_{k+1}^n)\big) + \prod_{i=1}^{p+1} s_5(d_{k+i}).
\end{aligned}$$

This is equation (28) for $p + 1$. It can be seen in exactly the same way that equation (29) is also valid for $p + 1$. But that means that equation (30) is correct for $p + 1$. So this will end the induction step and we find that (30) is true for all $p \geq 1$*. Consider the following guarded recursive specification E_6.

$$\begin{aligned}
E_6 = \big\{\, X_n^6 &= t \cdot Y_n^6 + \prod_{i=1}^{p} s_5(d_{k+i}) \\
Y_n^6 &= c_2(\tilde{d}_{k+1}) \cdot X_{n+1}^6 + \prod_{i=1}^{p} s_5(d_{k+i}) \;\big|\; n \geq 0 \,\big\}.
\end{aligned}$$

With the aid of equations (24) and (25) we can conclude, using equation (30) that, if we put for all $n \geq 0$

$$\begin{aligned}
X_n^6 &= \pi_J^p \circ \beta_1 \circ \partial_H\big(D_k(\tilde{d}_{k+1}^n)\big) \\
Y_n^6 &= \pi_J^p \circ \beta_1 \circ \partial_H\big(E_k(\tilde{d}_{k+1}^n)\big)
\end{aligned}$$

* Notice that now equations (28) and (29) are true for all $p \geq 1$, as well.

this is a solution for E_6. With a collapsing-argument we find that

$$\left.\begin{array}{l}\pi_J^p \circ \tau_I \circ \partial_H\big(D_k(\tilde{d}_{k+1}^n)\big)\\ \pi_J^p \circ \tau_I \circ \partial_H\big(E_k(\tilde{d}_{k+1}^n)\big)\end{array}\right\} = \tau \cdot \prod_{i=1}^{p} s_5(d_{k+i}). \tag{36}$$

Consider the guarded recursive specification E_7.

$$\begin{aligned} E_7 = \Big\{\, X_n^7 &= c_2(\tilde{d}_{k+1}) \cdot Y_{n+1}^7 + c_3(\tilde{d}_{k+1}) \cdot \prod_{i=1}^{p} s_5(d_{k+i}) \\ Y_{n+1}^7 &= t \cdot X_{n+1}^7 + c_3(\tilde{d}_{k+1}) \cdot \prod_{i=1}^{p} s_5(d_{k+i}) \;\Big|\; n \geq 1 \,\Big\}. \end{aligned}$$

It is easy to see with the aid of equations (20), (22) and (36) that, if we put for all $n \geq 1$

$$\begin{aligned} X_n^7 &= \pi_J^p \circ \beta_4 \circ \partial_H\big(B_k(\tilde{d}_{k+1}^n)\big) \\ Y_{n+1}^7 &= \pi_J^p \circ \beta_4 \circ \partial_H\big(C_k(\tilde{d}_{k+1}^{n+1})\big), \end{aligned}$$

this will solve the system E_7. So, in particular, we find, using a collapsing-argument, for all $n \geq 1$

$$\pi_J^p \circ \tau_I \circ \partial_H\big(C_k(\tilde{d}_{k+1}^n)\big) = \tau \cdot \prod_{i=1}^{p} s_5(d_{k+i}). \tag{37}$$

Now we can find with the aid of equation (17), (19) and (37) that for $k = 0$

$$\begin{aligned} \pi_J^p \circ \tau_I \circ \partial_H\big(A_0(\lambda)\big) &= \tau \cdot \pi_J^p \circ \tau_I \circ \partial_H\big(B_0(\lambda)\big) \\ &= \tau \cdot \pi_J^p \circ \tau_I \circ \partial_H\big(C_0(\tilde{d}_1)\big) \\ &= \tau \cdot \prod_{i=1}^{p} s_5(d_i). \end{aligned}$$

Thus, we have proved for all $p \geq 1$

$$\pi_J^p \circ \tau_I \circ \partial_H\big(A_0(\lambda)\big) = \tau \cdot \pi_J^p\Big(\prod_{i=1}^{\infty} s_5(d_i)\Big)$$

and with a *GIP* argument we can conclude the proof.

Remark (4.4.9)

In this verification, we have seen the frequent use of a so-called collapsing-argument. We can imagine that if the protocol becomes more complicated the collapsing-argument will become more complicated, as well. We can provide for each situation a new collapsing-argument; but it will be better to formulate a general theorem on collapsing-arguments that we can use in every situation. Fortunately, such a theorem already exists. Hereinafter, we will enumerate all the definitions needed in order to state this theorem.

The following definitions are taken from [**40**].

Definition (4.4.10)

Let $E = \{x = t_x(X) : x \in X\}$ be a guarded recursive specification (without abstracting operators). Let $I \subseteq A$. We will call a subset $C \subseteq X$ a cluster of I in E if $C \neq \emptyset$ and for all $x \in C$ we have $i_1, \ldots, i_m \in I \cup \{\tau\}$, $x_1, \ldots, x_m \in C$ and $y_1, \ldots, y_n \in X \setminus C$ $(n \geq 1, m \geq 0)$ such that the equation for x in the guarded recursive specification E is

$$x = \sum_{k=1}^{m} i_k \cdot x_k + \sum_{l=1}^{n} y_l,$$

in which the last summand is omitted if $n = 0$. We will call the variables y_l exits of x and we write $U(x) = \{y_1, \ldots, y_n\}$. We will use $U(C) = \bigcup_{x \in C} U(x)$ for the exit set of the cluster C. We will call a cluster *conservative* if every exit $y \in U(C)$ is accessible from every variable in the cluster by doing zero or more steps from $I \cup \{\tau\}$ in the cluster to a cluster-variable which has exit y.

Definition (4.4.11) *Cluster Fair Abstraction Rule*

Let E be a guarded recursive specification and let $I \subseteq A$ with $|I| \geq 2$. Let C be a conservative cluster of E in I. Let $|U(C)| < \infty$. Then we have for all $x \in C$:

$$\tau_I(x) = \tau \cdot \sum_{y \in U(C)} \tau_I(y).$$

We will refer to this rule with the abbreviation *CFAR*$^\infty$, since it is not supposed that the cluster itself is finite; only the exit set of the cluster. If C is a finite set then the abbreviation *CFAR* will be used.

The following theorem is taken from [**40**]. Only some notations are modified.

Theorem (4.4.12)

$$ACP_{\tau,u} + RDP + RSP + ODP + OSP + KFAR_1 + GIP \vdash CFAR^\infty.$$

Example (4.4.13)

Let us take a look at the guarded recursive specification E_1 that can be found in the proof of theorem (4.4.8).

$$\begin{aligned} E_1 = \{\, X_n^1 &= t \cdot Y_n^1 + Z \\ Y_n^1 &= c_2(\tilde{d}_{k+2}) \cdot X_{n+1}^1 + Z \\ Z &= \prod_{i=1}^{p} s_5(d_{k+1+i}) \;\Big|\; n \geq 0 \,\}. \end{aligned}$$

It is not very hard to see that $C = \{X_n^1, Y_n^1 : n \geq 0\}$ is a conservative cluster of E_1 in I and that for the set of exits we have $U(C) = \{Z\}$. Hence, we may apply *CFAR*$^\infty$ and we find thus for all $n \geq 0$

$$\tau_I(X_n^1) = \tau_I(Y_n^1) = \tau \cdot \tau_I(Z) = \tau \cdot \prod_{i=1}^{p} s_5(d_{k+1+i}).$$

So, in fact, our collapsing-argument is just a proof of theorem (4.4.12) in a special case.

In our next example we will make things a little less unrealistic. We will change the behaviour of the queue Q that already appeared in the specification of our alternating bit protocol: the queue can lose its last frame. We will express this with an atomic action l (for *lost*) as follows.

$$Q(\lambda) = \sum_{x \in E} r_2(x) \cdot Q(x)$$

$$Q(x * \sigma) = \sum_{y \in E} r_2(y) \cdot Q(x * \sigma * y) + s_3(x) \cdot Q(\sigma) + l \cdot Q(\sigma).$$

All the other components of our alternating bit protocol will remain the same and will be used in the following theorem. Observe that we use the same symbol for this faulty queue. This is done because abbreviations in the former theorem can be used again. We will enumerate the following sets that will be used in this theorem.

$$\begin{aligned} H = &\left\{\, r_1(d), s_1(d) : d \in D \right\} \\ &\cup \left\{\, r_2(x), r_3(x), s_2(x), s_3(x) : x \in E \right\} \\ &\cup \left\{\, r_4(\overline{k}), s_4(\overline{k}) : \overline{k} \in \mathbb{Z}/2\mathbb{Z} \,\right\} \\ I = &\left\{\, c_1(d) : d \in D \,\right\} \\ &\cup \left\{\, c_2(x), c_3(x) : x \in E \,\right\} \\ &\cup \left\{\, c_4(\overline{k}) : \overline{k} \in \mathbb{Z}/2\mathbb{Z} \,\right\} \cup \{l, t\}. \end{aligned}$$

Notice that the only difference with the sets used in theorem (4.4.8) is the extra l in the abstraction set I.

Theorem (4.4.14)

Let Q be the above queue with the possibility that the last frame can be lost. The other components are the same that appear in theorem (4.4.8).

$$\tau_I \circ \partial_H \Big(S(\overline{0}) \parallel Q(\lambda) \parallel R(\overline{0}) \parallel \prod_{i=1}^{\infty} s_1(d_i) \Big) = \tau \cdot \prod_{i=1}^{\infty} s_5(d_i).$$

Proof. The structure of this proof will be the same as before. We will first calculate

$$\partial_H \Big(S(\overline{0}) \parallel Q(\lambda) \parallel R(\overline{0}) \parallel \prod_{i=1}^{\infty} s_1(d_i) \Big).$$

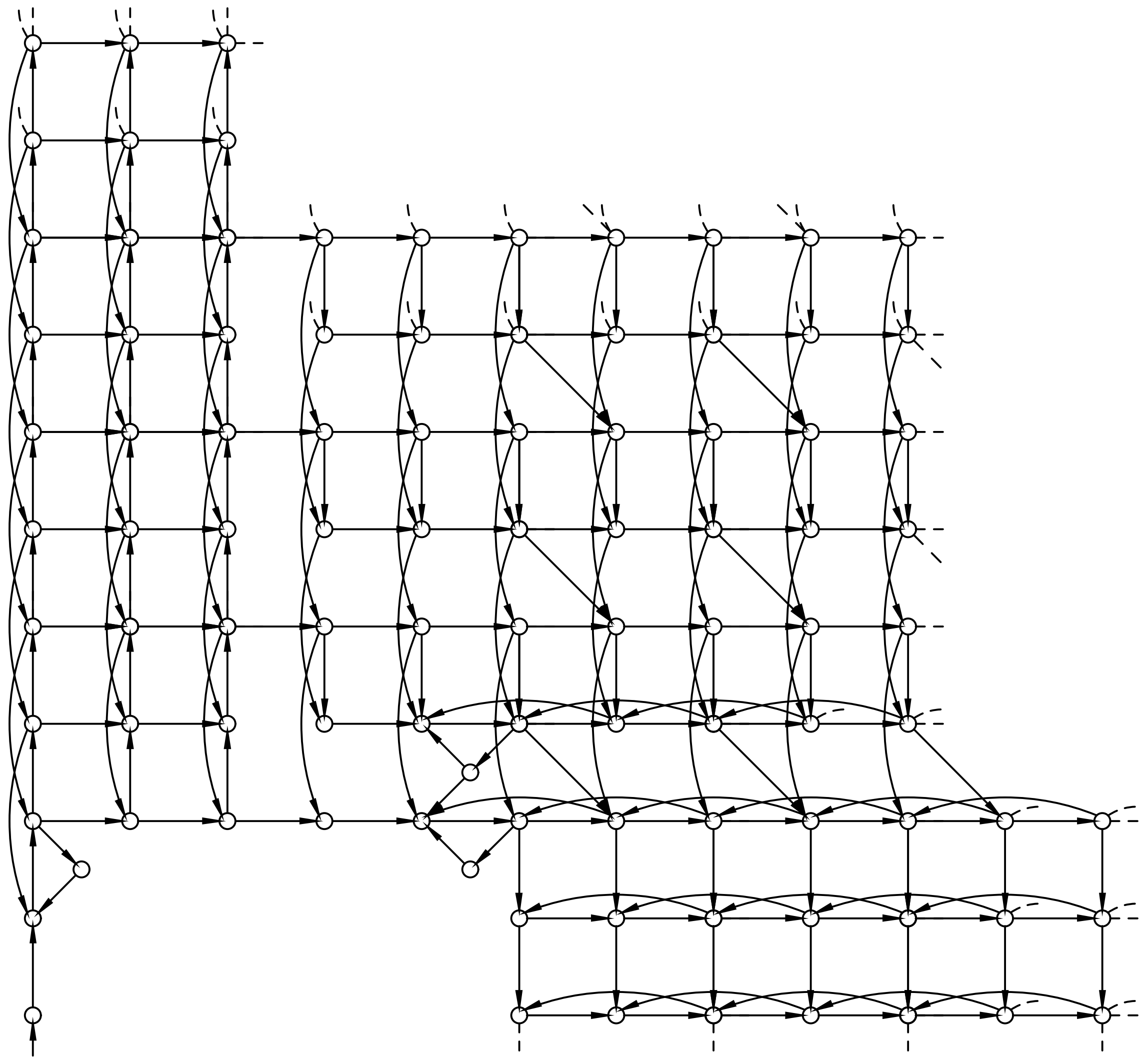

Figure 4.9. A state transition diagram.

We have included a picture of this process in figure 4.9 on page 201. The actual calculation of this process will be left to the reader, but we will provide some explanation such that with the aid of the picture and the specification it will be not that hard to verify that the result is correct. We will use the same notations as in theorem (4.4.8). In order to give the specification we will need three more abbreviations.

$$H_k(\sigma) = S(\tilde{d}_{k+1}) \parallel Q(\sigma) \parallel SA(\overline{k}) \parallel \prod_{i=k+2}^{\infty} s_1(d_i)$$

$$I_k(\sigma) = S(\overline{k+1}) \parallel Q(\sigma) \parallel SA(\overline{k}) \parallel \prod_{i=k+2}^{\infty} s_1(d_i)$$

$$J_k(\sigma) = S(\tilde{d}_{k+2}) \parallel Q(\sigma) \parallel SA(\overline{k}) \parallel \prod_{i=k+3}^{\infty} s_1(d_i)$$

Hereinafter, we will enumerate the guarded recursive specification. We have $k \geq 0$ and $n \geq 1$. For m we will mention its range explicitly. Then we will discuss this result.

$$\partial_H\big(A_k(\lambda)\big) = c_1(d_{k+1}) \cdot \partial_H\big(B_k(\lambda)\big) \tag{38}$$

$$\begin{aligned}\partial_H\big(A_{k+1}(\tilde{d}^n_{k+1})\big) = {} & l \cdot \partial_H\big(A_{k+1}(\tilde{d}^{n-1}_{k+1})\big) + c_1(d_{k+2}) \cdot \partial_H\big(B_{k+1}(\tilde{d}^n_{k+1})\big) \\ & + c_3(\tilde{d}_{k+1}) \cdot \partial_H\big(I_k(\tilde{d}^{n-1}_{k+1})\big)\end{aligned} \tag{39}$$

$$\partial_H\big(B_k(\lambda)\big) = c_2(\tilde{d}_{k+1}) \cdot \partial_H\big(C_k(\tilde{d}_{k+1})\big) \tag{40}$$

$$\begin{aligned}\partial_H\big(B_k(\tilde{d}^n_{k+1})\big) = {} & l \cdot \partial_H\big(B_k(\tilde{d}^{n-1}_{k+1})\big) + c_2(\tilde{d}_{k+1}) \cdot \partial_H\big(C_k(\tilde{d}^{n+1}_{k+1})\big) \\ & + c_3(\tilde{d}_{k+1}) \cdot \partial_H\big(E_k(\tilde{d}^{n-1}_{k+1})\big)\end{aligned} \tag{41}$$

$$\begin{aligned}\partial_H\big(B_{k+1}(\tilde{d}^n_{k+1} * \tilde{d}^m_{k+2})\big) = {} & l \cdot \partial_H\big(B_{k+1}(\tilde{d}^{n-1}_{k+1} * \tilde{d}^m_{k+2})\big) \qquad (m \geq 0) \\ & + c_3(\tilde{d}_{k+1}) \cdot \partial_H\big(J_k(\tilde{d}^{n-1}_{k+1} * \tilde{d}^m_{k+2})\big) \\ & + c_2(\tilde{d}_{k+2}) \cdot \partial_H\big(C_{k+1}(\tilde{d}^n_{k+1} * \tilde{d}^{m+1}_{k+2})\big)\end{aligned} \tag{42}$$

$$\partial_H\big(C_k(\lambda)\big) = t \cdot \partial_H\big(B_k(\lambda)\big) \tag{43}$$

$$\begin{aligned}\partial_H\big(C_k(\tilde{d}^n_{k+1})\big) = {} & l \cdot \partial_H\big(C_k(\tilde{d}^{n-1}_{k+1})\big) + t \cdot \partial_H\big(B_k(\tilde{d}^n_{k+1})\big) \\ & + c_3(\tilde{d}_{k+1}) \cdot \partial_H\big(D_k(\tilde{d}^{n-1}_{k+1})\big)\end{aligned} \tag{44}$$

$$\begin{aligned}\partial_H\big(C_{k+1}(\tilde{d}^n_{k+1} * \tilde{d}^m_{k+2})\big) = {} & l \cdot \partial_H\big(C_{k+1}(\tilde{d}^{n-1}_{k+1} * \tilde{d}^m_{k+2})\big) \qquad (m \geq 1) \\ & + c_3(\tilde{d}_{k+1}) \cdot \partial_H\big(G_k(\tilde{d}^{n-1}_{k+1} * \tilde{d}^m_{k+2})\big) \\ & + t \cdot \partial_H\big(B_{k+1}(\tilde{d}^n_{k+1} * \tilde{d}^m_{k+2})\big)\end{aligned} \tag{45}$$

$$\partial_H\big(D_k(\lambda)\big) = t \cdot \partial_H\big(E_k(\lambda)\big) + s_5(d_{k+1}) \cdot \partial_H\big(F_k(\lambda)\big) \tag{46}$$

$$\begin{aligned}\partial_H\big(D_k(\tilde{d}^n_{k+1})\big) = {} & l \cdot \partial_H\big(D_k(\tilde{d}^{n-1}_{k+1})\big) + t \cdot \partial_H\big(E_k(\tilde{d}^n_{k+1})\big) \\ & + s_5(d_{k+1}) \cdot \partial_H\big(F_k(\tilde{d}^n_{k+1})\big)\end{aligned} \tag{47}$$

$$\begin{aligned}\partial_H\big(E_k(\lambda)\big) = {} & c_2(\tilde{d}_{k+1}) \cdot \partial_H\big(D_k(\tilde{d}_{k+1})\big) \\ & + s_5(d_{k+1}) \cdot \partial_H\big(H_k(\lambda)\big)\end{aligned} \tag{48}$$

$$\begin{aligned}\partial_H\big(E_k(\tilde{d}^n_{k+1})\big) = {} & l \cdot \partial_H\big(E_k(\tilde{d}^{n-1}_{k+1})\big) + c_2(\tilde{d}_{k+1}) \cdot \partial_H\big(D_k(\tilde{d}^{n+1}_{k+1})\big) \\ & + s_5(d_{k+1}) \cdot \partial_H\big(H_k(\tilde{d}^n_{k+1})\big)\end{aligned} \tag{49}$$

$$\partial_H\big(F_k(\lambda)\big) = t \cdot \partial_H\big(H_k(\lambda)\big) + c_4(\overline{k}) \cdot \partial_H\big(A_{k+1}(\lambda)\big) \tag{50}$$

$$\begin{aligned}\partial_H\big(F_k(\tilde{d}^n_{k+1})\big) = {} & l \cdot \partial_H\big(F_k(\tilde{d}^{n-1}_{k+1})\big) + t \cdot \partial_H\big(H_k(\tilde{d}^n_{k+1})\big) \\ & + c_4(\overline{k}) \cdot \partial_H\big(A_{k+1}(\tilde{d}^n_{k+1})\big)\end{aligned} \tag{51}$$

$$\partial_H\big(G_k(\lambda)\big) = t \cdot \partial_H\big(J_k(\lambda)\big) + c_4(\overline{k}) \cdot \partial_H\big(B_{k+1}(\lambda)\big) \tag{52}$$

$$\partial_H\big(G_k(\tilde{d}^m_{k+2})\big) = l \cdot \partial_H\big(G_k(\tilde{d}^{m-1}_{k+2})\big) + t \cdot \partial_H\big(J_k(\tilde{d}^m_{k+2})\big) + c_4(\overline{k}) \cdot \partial_H\big(B_{k+1}(\tilde{d}^m_{k+2})\big) \quad (m \geq 1) \tag{53}$$

$$\partial_H\big(G_k(\tilde{d}^n_{k+1} * \tilde{d}^m_{k+2})\big) = l \cdot \partial_H\big(G_k(\tilde{d}^{n-1}_{k+1} * \tilde{d}^m_{k+2})\big) + t \cdot \partial_H\big(J_k(\tilde{d}^n_{k+1} * \tilde{d}^m_{k+2})\big) + c_4(\overline{k}) \cdot \partial_H\big(B_{k+1}(\tilde{d}^n_{k+1} * \tilde{d}^m_{k+2})\big) \quad (m \geq 1) \tag{54}$$

$$\partial_H\big(H_k(\lambda)\big) = c_2(\tilde{d}_{k+1}) \cdot \partial_H\big(F_k(\tilde{d}_{k+1})\big) \tag{55}$$

$$\partial_H\big(H_k(\tilde{d}^n_{k+1})\big) = l \cdot \partial_H\big(H_k(\tilde{d}^{n-1}_{k+1})\big) + c_2(\tilde{d}_{k+1}) \cdot \partial_H\big(F_k(\tilde{d}^{n+1}_{k+1})\big) \tag{56}$$

$$\partial_H\big(I_k(\lambda)\big) = c_1(d_{k+2}) \cdot \partial_H\big(J_k(\lambda)\big) \tag{57}$$

$$\partial_H\big(I_k(\tilde{d}^n_{k+1})\big) = l \cdot \partial_H\big(I_k(\tilde{d}^{n-1}_{k+1})\big) + c_1(d_{k+2}) \cdot \partial_H\big(J_k(\tilde{d}^n_{k+1})\big) \tag{58}$$

$$\partial_H\big(J_k(\lambda)\big) = c_2(\tilde{d}_{k+2}) \cdot \partial_H\big(G_k(\tilde{d}_{k+2})\big) \tag{59}$$

$$\partial_H\big(J_k(\tilde{d}^m_{k+2})\big) = l \cdot \partial_H\big(J_k(\tilde{d}^{m-1}_{k+2})\big) + c_2(\tilde{d}_{k+2}) \cdot \partial_H\big(G_{k+1}(\tilde{d}^{m+1}_{k+2})\big) \quad (m \geq 1) \tag{60}$$

$$\partial_H\big(J_k(\tilde{d}^n_{k+1} * \tilde{d}^m_{k+2})\big) = l \cdot \partial_H\big(J_k(\tilde{d}^{n-1}_{k+1} * \tilde{d}^m_{k+2})\big) + c_2(\tilde{d}_{k+2}) \cdot \partial_H\big(G_{k+1}(\tilde{d}^n_{k+1} * \tilde{d}^{m+1}_{k+2})\big) \quad (m \geq 0) \tag{61}$$

At this point, we will clarify the guarded recursive specification consisting of the equations (38)–(61). We are to give a guarded recursive specification of the process $\partial_H\big(A_0(\lambda)\big)$. If we calculate this, we can see in figure 4.9, starting from the root of the graph, that we will rise to the next node, which is $\partial_H\big(B_0(\lambda)\big)$. This part of the picture corresponds to equation (38) for $k = 0$. If we calculate this node we will go up again and we will enter "the single* *BC*-trace", which corresponds to equations (41) and (44). We can enter a triangle from this node that corresponds to equations (44) for $n = 1$ and equation (43). If we go one step to the right from the single *BC*-trace we will enter the *DE*-trace, which corresponds to the equations (46)–(49). In the *DE*-trace we can do an $s_5(d_0)$ step to the right and then we will enter the *FH*-trace. This trace resembles the equations (50), (51), (55) and (56). If we step to the right in the *FH*-trace on the lowest level, (see equation (50)), we see that we will enter $\partial_H\big(A_1(\lambda)\big)$, via an acknowledgement $c_4(\overline{0})$. This corresponds to equation (38) for $k = 1$. Thereupon, we will enter the single *BC*-trace for $k = 1$, or a triangle, and so on. If we do a step to the right from the *FH*-trace on another level, we will enter an *A*-node, which corresponds to equation (39) (confer (51)). At this *A*-node we have three choices: we can jump to a lower *A*-node, we can go down to an *I*-node (57) and (58), or we can enter a double** *BC*-trace, which corresponds to

* Single stands for the fact that there is only one type of elements in the queue.

** Double stands for the fact that there are two types of elements in the queue.

the equations (42) and (45). From the double BC-trace we can do a step down and enter the JG-trace or jump down to a lower BC-trace. This can be a double BC-trace, or if we are on the first level, a single one. From the I-node we can jump to a lower I-node, except on the first level, or we can enter the JG-trace, as well. This JG-trace corresponds to equations (52)–(54) and (59)–(61). From the JG-trace we can jump down to a lower JG-trace, except on the first level; there we can only jump to the left in the same JG-trace. Or we can do a "skewed" step down to a lower BC-trace, which can be both double or single. There is one node that we have not met yet: it is the node between the lowest JG-trace and the single BC-trace right under it. This node corresponds to equation (52).

Now we have seen how this large guarded recursive specification can be deduced, we will briefly discuss the overall structure of the sequel: first, we will make an assumption for one $p \geq 1$ on a DE-trace, for which we already know that it holds for $p = 1$. With this assumption we can derive a conclusion for an FH-trace. To achieve this we will need an intermediary result that we will prove with induction. Once we derived the conclusion on this FH-trace we will set off a chain reaction: we know that the assumption on the DE-trace holds for $p = 1$ and then we conclude that the conclusion on the FH-trace is valid for $p = 1$. With this knowledge we can derive that the assumption on the DE-trace is correct for $p = 2$ and so we find that the conclusion concerning the FH-trace holds for $p = 2$ et cetera. This will be done with induction on p. In the end it will turn out that both the assumption and the conclusion hold for all $p \geq 1$. This fact will be used to make a small calculation on a single BC-trace. Using this together with a GIP argument, we can conclude the proof of 4.4.14.

Let $J = \{s_5(d) : d \in D\}$. Let $\beta_1 = \beta(I, J)$. Let, for the moment, $p \geq 1$ be fixed. Suppose that the following holds. For all $k \geq 0$ and for all $n \geq 1$:

$$\pi_J^p \circ \beta_1 \circ \partial_H\big(D_k(\lambda)\big) = t \cdot \pi_J^p \circ \beta_1 \circ \partial_H\big(E_k(\lambda)\big) + \prod_{i=1}^{p} s_5(d_{k+i}) \tag{62}$$

$$\begin{aligned} \pi_J^p \circ \beta_1 \circ \partial_H\big(D_k(\tilde{d}_{k+1}^n)\big) &= t \cdot \pi_J^p \circ \beta_1 \circ \partial_H\big(E_k(\tilde{d}_{k+1}^n)\big) \\ &\quad + l \cdot \pi_J^p \circ \beta_1 \circ \partial_H\big(D_k(\tilde{d}_{k+1}^{n-1})\big) + \prod_{i=1}^{p} s_5(d_{k+i}) \end{aligned} \tag{63}$$

$$\begin{aligned} \pi_J^p \circ \beta_1 \circ \partial_H\big(E_k(\lambda)\big) &= c_2(\tilde{d}_{k+1}) \cdot \pi_J^p \circ \beta_1 \circ \partial_H\big(D_k(\tilde{d}_{k+1})\big) \\ &\quad + \prod_{i=1}^{p} s_5(d_{k+i}) \end{aligned} \tag{64}$$

$$\pi_J^p \circ \beta_1 \circ \partial_H\big(E_k(\tilde{d}_{k+1}^n)\big) = c_2(\tilde{d}_{k+1}) \cdot \pi_J^p \circ \beta_1 \circ \partial_H\big(D_k(\tilde{d}_{k+1}^{n+1})\big) \tag{65}$$

$$+ l \cdot \pi_J^p \circ \beta_1 \circ \partial_H\big(E_k(\tilde{d}_{k+1}^{n-1})\big) + \prod_{i=1}^{p} s_5(d_{k+i}).$$

Then the following can be derived from this assumption. For all $k, n \geq 0$:

$$\left.\begin{array}{l} \pi_J^p \circ \tau_I \circ \partial_H\big(F_k(\tilde{d}_{k+1}^n)\big) \\ \pi_J^p \circ \tau_I \circ \partial_H\big(H_k(\tilde{d}_{k+1}^n)\big) \end{array}\right\} = \tau \cdot \prod_{i=1}^{p} s_5(d_{k+1+i}). \tag{66}$$

To prove this we will first deduce from the assumptions that we made an intermediary result on the *JG*-traces and the double *BC*-traces. Hereinafter, we will state it and we will prove it with induction on u. For all $k, u, m \geq 0$:

$$\left.\begin{array}{l} \pi_J^p \circ \tau_I \circ \partial_H\big(B_{k+1}(\tilde{d}_{k+1}^u * \tilde{d}_{k+2}^m)\big) \\ \pi_J^p \circ \tau_I \circ \partial_H\big(C_{k+1}(\tilde{d}_{k+1}^u * \tilde{d}_{k+2}^{m+1})\big) \\ \pi_J^p \circ \tau_I \circ \partial_H\big(G_k(\tilde{d}_{k+1}^u * \tilde{d}_{k+2}^{m+1})\big) \\ \pi_J^p \circ \tau_I \circ \partial_H\big(J_k(\tilde{d}_{k+1}^u * \tilde{d}_{k+2}^m)\big) \end{array}\right\} = \tau \cdot \prod_{i=1}^{p} s_5(d_{k+1+i}) \tag{67}$$

As we already announced, we will prove (67) with induction on u. So let first of all $u = 0$ and fix $k \geq 0$. Consider the following guarded recursive specification E_1.

$$\begin{aligned} E_1 = \{\, X_0^1 &= t \cdot Y_0^1 + Z_1 \\ Y_0^1 &= c_2(\tilde{d}_{k+2}) \cdot X_1^1 + Z_1 \\ X_n^1 &= l \cdot X_{n-1}^1 + t \cdot Y_n^1 + Z_1 \\ Y_n^1 &= l \cdot Y_{n-1}^1 + c_2(\tilde{d}_{k+2}) \cdot X_{n+1}^1 + Z_1 \\ Z_1 &= \prod_{i=1}^{p} s_5(d_{k+1+i}) \;\big|\; n \geq 1 \,\}. \end{aligned}$$

Let $C_1 = \{X_n^1, Y_n^1 : n \geq 0\}$. Then C_1 is a conservative cluster of E_1 in I. Since the set of exits $U(C_1) = \{Z_1\}$ is finite, we may apply $CFAR^\infty$ and we find for all $n \geq 0$:

$$\tau_I(X_n^1) = \tau_I(Y_n^1) = \tau \cdot \prod_{i=1}^{p} s_5(d_{k+1+i}).$$

Because of equations (62)–(65) we see that if we put for all $n \geq 0$

$$\begin{aligned} X_n^1 &= \pi_J^p \circ \beta_1 \circ \partial_H\big(D_{k+1}(\tilde{d}_{k+2}^n)\big) \\ Y_n^1 &= \pi_J^p \circ \beta_1 \circ \partial_H\big(E_{k+1}(\tilde{d}_{k+2}^n)\big), \end{aligned}$$

this will solve the guarded recursive specification E_1. Now consider the following calculation.

$$\begin{aligned}
\tau \cdot \prod_{i=1}^{p} s_5(d_{k+1+i}) &= \tau_I(X_n^1) \\
&= \tau_I \circ \pi_J^p \circ \beta_1 \circ \partial_H\big(D_{k+1}(\tilde{d}_{k+2}^n)\big) \\
&= \pi_J^p \circ \tau_I \circ \beta_1 \circ \partial_H\big(D_{k+1}(\tilde{d}_{k+2}^n)\big) \quad \text{use (4.3.12)} \\
&= \pi_J^p \circ \tau_I \circ \partial_H\big(D_{k+1}(\tilde{d}_{k+2}^n)\big) \text{ see lemma (4.4.6)}
\end{aligned}$$

Such a calculation can be made for E_{k+1}, too. We find thus for all $n \geq 0$

$$\left.\begin{array}{l}
\pi_J^p \circ \tau_I \circ \partial_H\big(D_{k+1}(\tilde{d}_{k+2}^n)\big) \\
\pi_J^p \circ \tau_I \circ \partial_H\big(E_{k+1}(\tilde{d}_{k+2}^n)\big)
\end{array}\right\} = \tau \cdot \prod_{i=1}^{p} s_5(d_{k+1+i}). \tag{68}$$

Now consider the following guarded recursive specification E_2.

$$\begin{aligned}
E_2 = \{\ X_0^2 &= c_2(\tilde{d}_{k+2}) \cdot Y_1^2 \\
Y_0^2 &= t \cdot X_0^2 \\
X_n^2 &= l \cdot X_{n-1}^2 + c_2(\tilde{d}_{k+2}) \cdot Y_{n+1}^2 + Z_2 \\
Y_n^2 &= l \cdot Y_{n-1}^2 + t \cdot X_n^2 + Z_2 \\
Z_2 &= c_3(\tilde{d}_{k+2}) \cdot \prod_{i=1}^{p} s_5(d_{k+1+i}) \ \big|\ n \geq 1\,\}.
\end{aligned}$$

Let $C_2 = \{X_n^2, Y_n^2 : n \geq 0\}$. Then C_2 is a conservative cluster of E_2 in I. Since the set of exits $U(C_2) = \{Z_2\}$ is finite, we may apply $CFAR^\infty$ and we find for all $n \geq 0$:

$$\tau_I(X_n^2) = \tau_I(Y_n^2) = \tau \cdot \prod_{i=1}^{p} s_5(d_{k+1+i}).$$

Let $\beta_2 = \beta\big(I, \{c_3(\tilde{d}_{k+2})\}\big)$. Hereinafter, we will show that if we put for all $n \geq 0$

$$\begin{aligned}
X_n^2 &= \pi_J^p \circ \beta_2 \circ \partial_H\big(B_{k+1}(\tilde{d}_{k+2}^n)\big) \\
Y_n^2 &= \pi_J^p \circ \beta_2 \circ \partial_H\big(C_{k+1}(\tilde{d}_{k+2}^n)\big),
\end{aligned} \tag{69}$$

this will solve the guarded recursive specification E_2. For $n = 0$ it is trivial to see that, with the aid of equations (40) and (43), this is true for the first two equations of E_2. Now let $n \geq 1$. We will verify the third equation of E_2.

$$\begin{aligned}
\pi_J^p \circ \beta_2 \circ \partial_H\big(B_{k+1}(\tilde{d}_{k+2}^n)\big) \overset{(41)}{=} & \ l \cdot \pi_J^p \circ \beta_2 \circ \partial_H\big(B_{k+1}(\tilde{d}_{k+2}^{n-1})\big) \\
& + c_2(\tilde{d}_{k+2}) \cdot \pi_J^p \circ \beta_2 \circ \partial_H\big(C_{k+1}(\tilde{d}_{k+2}^{n+1})\big)
\end{aligned}$$

$$
\begin{aligned}
&+ c_3(\tilde{d}_{k+2}) \cdot \pi_J^p \circ \tau_I \circ \partial_H\big(E_{k+1}(\tilde{d}_{k+2}^{n-1})\big) \\
\overset{(68)}{=} &\, l \cdot \pi_J^p \circ \beta_2 \circ \partial_H\big(B_{k+1}(\tilde{d}_{k+2}^{n-1})\big) \\
&+ c_2(\tilde{d}_{k+2}) \cdot \pi_J^p \circ \beta_2 \circ \partial_H\big(C_{k+1}(\tilde{d}_{k+2}^{n+1})\big) \\
&+ c_3(\tilde{d}_{k+2}) \cdot \tau \cdot \prod_{i=1}^{p} s_5(d_{k+1+i}).
\end{aligned}
$$

So we see that this solves the equation in E_2 for X_n^2. A similar calculation will yield the same result for the fourth equation of E_2. So the claim that (69) is a solution for E_2 is valid. We find

$$
\begin{aligned}
\tau \cdot \prod_{i=1}^{p} s_5(d_{k+1+i}) &= \tau_I(X_n^2) \\
&= \tau_I \circ \pi_J^p \circ \beta_2 \circ \partial_H\big(B_{k+1}(\tilde{d}_{k+2}^{n})\big) \\
&= \pi_J^p \circ \tau_I \circ \beta_2 \circ \partial_H\big(B_{k+1}(\tilde{d}_{k+2}^{n})\big) && \text{use (4.3.12)} \\
&= \pi_J^p \circ \tau_I \circ \partial_H\big(B_{k+1}(\tilde{d}_{k+2}^{n})\big). && \text{see (4.4.6)}
\end{aligned}
$$

Such a calculation can be made for C_{k+1}, too. Thus, we have for all $n \geq 0$

$$
\left.\begin{aligned}
\pi_J^p \circ \tau_I \circ \partial_H\big(B_{k+1}(\tilde{d}_{k+2}^{n})\big) \\
\pi_J^p \circ \tau_I \circ \partial_H\big(C_{k+1}(\tilde{d}_{k+2}^{n})\big)
\end{aligned}\right\} = \tau \cdot \prod_{i=1}^{p} s_5(d_{k+1+i}). \tag{70}
$$

This means that the first two equations of (67) are proved for $u = 0$ and our fixed $k \geq 0$. Consider the next guarded recursive specification E_3.

$$
\begin{aligned}
E_3 = \big\{\, X_0^3 &= t \cdot Y_0^3 + Z_3 \\
Y_0^3 &= c_2(\tilde{d}_{k+2}) \cdot X_1^3 \\
X_m^3 &= l \cdot X_{m-1}^3 + t \cdot Y_m^3 + Z_3 \\
Y_m^3 &= l \cdot Y_{m-1}^3 + c_2(\tilde{d}_{k+2}) \cdot X_{m+1}^3 \\
Z_3 &= c_4(\overline{k}) \cdot \prod_{i=1}^{p} s_5(d_{k+1+i}) \;\big|\; n \geq 1 \,\big\}.
\end{aligned}
$$

Let $C_3 = \{X_m^3, Y_m^3 : m \geq 0\}$. Then C_3 is a conservative cluster of E_3 in I. Since the set of exits $U(C_3) = \{Z_3\}$ is finite, we may apply *CFAR*$^\infty$ and we find for all $m \geq 0$:

$$
\tau_I(X_m^3) = \tau_I(Y_m^3) = \tau \cdot \prod_{i=1}^{p} s_5(d_{k+1+i}).
$$

Let $\beta_3 = \beta\big(I, \{c_4(\overline{k})\}\big)$. It will not be difficult to see that if we put for all $m \geq 0$

$$
\begin{aligned}
X_m^3 &= \pi_J^p \circ \beta_3 \circ \partial_H\big(G_k(\tilde{d}_{k+2}^{m})\big) \\
Y_m^3 &= \pi_J^p \circ \beta_3 \circ \partial_H\big(J_k(\tilde{d}_{k+2}^{m})\big),
\end{aligned} \tag{71}
$$

this will solve the system E_3. We will only verify the first equation for E_3. Consider thereto the following display.

$$\begin{aligned}\pi^p_J \circ \beta_3 \circ \partial_H\big(G_k(\lambda)\big) &\overset{(52)}{=} t \cdot \pi^p_J \circ \beta_3 \circ \partial_H\big(J_k(\lambda)\big) \\ &\qquad + c_4(\overline{k}) \cdot \pi^p_J \circ \tau_I \circ \partial_H\big(B_{k+1}(\lambda)\big) \\ &\overset{(70)}{=} t \cdot \pi^p_J \circ \beta_3 \circ \partial_H\big(J_k(\lambda)\big) \\ &\qquad + c_4(\overline{k}) \cdot \tau \cdot \prod_{i=1}^{p} s_5(d_{k+1+i}).\end{aligned}$$

The other equations can be handled in the same way while using equation (70). Thus, we find that (71) solves the guarded recursive specification E_3. When we use the fact that the generalized projection operator and the abstraction operator commute and the fact that the abstraction operator absorbs the selective abstraction operator β_3 we will find with a calculation that is similar to the ones we did for E_1 and E_2 that for all $m \geq 0$

$$\left.\begin{aligned}\pi^p_J \circ \tau_I \circ \partial_H\big(G_k(\tilde{d}^m_{k+2})\big) \\ \pi^p_J \circ \tau_I \circ \partial_H\big(J_k(\tilde{d}^m_{k+2})\big)\end{aligned}\right\} = \tau \cdot \prod_{i=1}^{p} s_5(d_{k+1+i}).$$

Now we find that the second two equations of (67) are also correct so we see that the basis of the induction is proved for (67), since $k \geq 0$ was arbitrarily chosen. Assume that (67) is valid for $u \geq 0$. We will prove it for $u+1$. Fix $k \geq 0$ and let E_4 be the following guarded recursive specification.

$$\begin{aligned}E_4 = \big\{\, X^4_m &= c_2(\tilde{d}_{k+2}) \cdot Y^4_{m+1} + Z_4 \\ Y^4_{m+1} &= t \cdot X^4_{m+1} + Z_4 \\ Z_4 &= \big(c_3(\tilde{d}_{k+1}) + l\big) \cdot \prod_{i=1}^{p} s_5(d_{k+1+i}) \;\big|\; m \geq 0 \,\big\}.\end{aligned}$$

Let $C_4 = \{X^4_m, Y^4_{m+1} : m \geq 0\}$. Then it is easy to see that C_4 is a conservative cluster of E_4 in I. The set of exits $U(C_4) = \{Z_4\}$ is finite, so we may apply $CFAR^\infty$ and we find for all $m \geq 0$

$$\tau_I(X^4_m) = \tau_I(Y^4_{m+1}) = \tau \cdot \prod_{i=1}^{p} s_5(d_{k+1+i}).$$

Let $\beta_4 = \beta\big(I, \{c_3(\tilde{d}_{k+1}), l\}\big)$. We will show that if we put for all $m \geq 0$

$$\begin{aligned}X^4_m &= \pi^p_J \circ \beta_4 \circ \partial_H\big(B_{k+1}(\tilde{d}^{u+1}_{k+1} * \tilde{d}^m_{k+2})\big) \\ Y^4_{m+1} &= \pi^p_J \circ \beta_4 \circ \partial_H\big(C_{k+1}(\tilde{d}^{u+1}_{k+1} * \tilde{d}^{m+1}_{k+2})\big),\end{aligned} \tag{72}$$

this will be a solution for the system E_4. For, contemplate the deduction below concerning the first equation of E_4.

$$\begin{aligned}
&\pi_J^p \circ \beta_4 \circ \partial_H\big(B_{k+1}(\tilde{d}_{k+1}^{u+1} * \tilde{d}_{k+2}^{m})\big)\\
&\quad\overset{(42)}{=} l \cdot \pi_J^p \circ \tau_I \circ \partial_H\big(B_{k+1}(\tilde{d}_{k+1}^{u} * \tilde{d}_{k+2}^{m})\big)\\
&\qquad + c_2(\tilde{d}_{k+2}) \cdot \pi_J^p \circ \beta_4 \circ \partial_H\big(C_{k+1}(\tilde{d}_{k+1}^{u+1} * \tilde{d}_{k+2}^{m+1})\big)\\
&\qquad + c_3(\tilde{d}_{k+1}) \cdot \pi_J^p \circ \tau_I \circ \partial_H\big(J_k(\tilde{d}_{k+1}^{u} * \tilde{d}_{k+2}^{m})\big)\\
&\quad\overset{(67)}{=} l \cdot \tau \cdot \prod_{i=1}^{p} s_5(d_{k+1+i}) + c_3(\tilde{d}_{k+1}) \cdot \tau \cdot \prod_{i=1}^{p} s_5(d_{k+1+i})\\
&\qquad + c_2(\tilde{d}_{k+2}) \cdot \pi_J^p \circ \beta_4 \circ \partial_H\big(C_{k+1}(\tilde{d}_{k+1}^{u+1} * \tilde{d}_{k+2}^{m+1})\big)\\
&\quad= c_2(\tilde{d}_{k+2}) \cdot \pi_J^p \circ \beta_4 \circ \partial_H\big(C_{k+1}(\tilde{d}_{k+1}^{u+1} * \tilde{d}_{k+2}^{m+1})\big)\\
&\qquad + \big(c_3(\tilde{d}_{k+1}) + l\big) \cdot \prod_{i=1}^{p} s_5(d_{k+1+i}).
\end{aligned}$$

Completely analogously to the above calculation we can deduce a similar result for the second equation of E_4. So we see that the solution for the guarded recursive specification E_4 is (72). We find easily using commutativity and absorption:

$$\left.\begin{array}{l}
\pi_J^p \circ \tau_I \circ \partial_H\big(B_{k+1}(\tilde{d}_{k+1}^{u+1} * \tilde{d}_{k+2}^{m})\big)\\
\pi_J^p \circ \tau_I \circ \partial_H\big(C_{k+1}(\tilde{d}_{k+1}^{u+1} * \tilde{d}_{k+2}^{m+1})\big)
\end{array}\right\} = \tau \cdot \prod_{i=1}^{p} s_5(d_{k+1+i}). \tag{73}$$

Thus we find that the first two equations of (67) are correct for $u+1$ for our chosen $k \geq 0$. To prove that the other two are also correct we will consider the following guarded recursive specification E_5.

$$\begin{aligned}
E_5 = \big\{\, X_{m+1}^5 &= t \cdot Y_{m+1}^5 + Z_1^5 + Z_2^5\\
Y_m^5 &= c_2(\tilde{d}_{k+2}) \cdot X_{m+1}^5 + Z_2^5\\
Z_1^5 &= c_4(\overline{k}) \cdot \prod_{i=1}^{p} s_5(d_{k+1+i})\\
Z_2^5 &= l \cdot \prod_{i=1}^{p} s_5(d_{k+1+i}) \;\big|\; m \geq 0 \,\big\}.
\end{aligned}$$

Let $C_5 = \{X_{m+1}^5, Y_m^5 : m \geq 0\}$. It is evident that C_5 is a conservative cluster of E_5 in I. The set of exits $U(C_5) = \{Z_1^5, Z_2^5\}$ is finite so in accordance with $CFAR^\infty$ we find for all $m \geq 0$

$$\tau_I(X_{m+1}^5) = \tau_I(Y_m^5) = \tau \cdot \prod_{i=1}^{p} s_5(d_{k+1+i}).$$

Let $\beta_5 = \beta\big(I, \{c_4(\overline{k}), l\}\big)$. Then it is easy to see with the aid of equations (54) and (61), the induction hypothesis, and the just derived result (73) that if we put for all $m \geq 0$

$$\begin{aligned} X^5_{m+1} &= \pi^p_J \circ \beta_5 \circ \partial_H\big(G_k(\tilde{d}^{u+1}_{k+1} * \tilde{d}^{m+1}_{k+2})\big) \\ Y^5_m &= \pi^p_J \circ \beta_5 \circ \partial_H\big(J_k(\tilde{d}^{u+1}_{k+1} * \tilde{d}^{m}_{k+2})\big), \end{aligned}$$

this will be a solution for the system E_5. It is deduced in exactly the same way as we did hereinbefore that

$$\left.\begin{aligned} &\pi^p_J \circ \tau_I \circ \partial_H\big(G_k(\tilde{d}^{u+1}_{k+1} * \tilde{d}^{m+1}_{k+2})\big) \\ &\pi^p_J \circ \tau_I \circ \partial_H\big(J_k(\tilde{d}^{u+1}_{k+1} * \tilde{d}^{m}_{k+2})\big) \end{aligned}\right\} = \tau \cdot \prod_{i=1}^{p} s_5(d_{k+1+i}).$$

Hence, we see that also the second two equations of (67) are valid for $u+1$, so this will end the induction step for our intermediate result since $k \geq 0$ was arbitrarily chosen.

At this point we will state three more guarded recursive specifications before we can conclude the proof of equation (66). The first matter that we are going to discuss is a result on the I-nodes. We want to calculate that for all $n \geq 0$

$$\pi^p_J \circ \tau_I \circ \partial_H\big(I_k(\tilde{d}^n_{k+1})\big) = \tau \cdot \prod_{i=1}^{p} s_5(d_{k+1+i}). \tag{74}$$

Therefore, we will consider the guarded recursive specification E_6.

$$\begin{aligned} E_6 = \Big\{\, X^6_0 &= c_1(d_{k+2}) \cdot \prod_{i=1}^{p} s_5(d_{k+1+i}) \\ X^6_n &= l \cdot X^6_{n-1} + X^6_0 \;\Big|\; n \geq 1 \,\Big\}. \end{aligned}$$

Let $C_6 = \{X^6_n : n \geq 1\}$. The set of exits is $U(C_6) = \{X^6_0\}$. It will be clear that C_6 is a conservative cluster for E_6 in I. Hence, according to *CFAR*$^\infty$ we find for all $n \geq 1$, which can be obtained for $n = 0$ directly:

$$\tau_I(X^6_0) = \tau_I(X^6_n) = \tau \cdot \prod_{i=1}^{p} s_5(d_{k+1+i}).$$

Let $\beta_6 = \beta\big(I, \{c_1(d_{k+2})\}\big)$. We will show that if we put for all $n \geq 0$

$$X^6_n = \pi^p_J \circ \beta_6 \circ \partial_H\big(I_k(\tilde{d}^n_{k+1})\big), \tag{75}$$

this will solve E_6. First, let $n = 0$ and consider the following.

$$\begin{aligned} \pi^p_J \circ \beta_6 \circ \partial_H\big(I_k(\lambda)\big) &\overset{(57)}{=} c_1(d_{k+2}) \cdot \pi^p_J \circ \tau_I \circ \partial_H\big(J_k(\lambda)\big) \\ &\overset{(67)}{=} c_1(d_{k+2}) \cdot \tau \cdot \prod_{i=1}^{p} s_5(d_{k+1+i}). \end{aligned}$$

Now we will handle the case $n \geq 1$.

$$\begin{aligned}\pi_J^p \circ \beta_6 \circ \partial_H\big(I_k(\tilde{d}_{k+1}^n)\big) &\overset{(58)}{=} l \cdot \pi_J^p \circ \beta_6 \circ \partial_H\big(I_k(\tilde{d}_{k+1}^{n-1})\big) \\ &\quad + c_1(d_{k+2}) \cdot \pi_J^p \circ \tau_I \circ \partial_H\big(J_k(\tilde{d}_{k+1}^n)\big) \\ &\overset{(67)}{=} l \cdot \pi_J^p \circ \beta_6 \circ \partial_H\big(I_k(\tilde{d}_{k+1}^{n-1})\big) \\ &\quad + c_1(d_{k+2}) \cdot \tau \cdot \prod_{i=1}^{p} s_5(d_{k+1+i}).\end{aligned}$$

So we see that (75) solves the guarded recursive specification E_6. Together with our calculations above we find that equation (74) is valid for all $n \geq 0$. The next thing that we are going to prove is something about the A-nodes. We will show that for all $n \geq 0$

$$\pi_J^p \circ \tau_I \circ \partial_H\big(A_{k+1}(\tilde{d}_{k+1}^n)\big) = \tau \cdot \prod_{i=1}^{p} s_5(d_{k+1+i}). \tag{76}$$

Consider the guarded recursive specification E_7 below.

$$\begin{aligned}E_7 = \Big\{\ X_0^7 &= c_1(d_{k+2}) \cdot \prod_{i=1}^{p} s_5(d_{k+1+i}) \\ X_n^7 &= l \cdot X_{n-1}^7 + X_0^7 + Z_7 \\ Z_7 &= c_3(\tilde{d}_{k+1}) \cdot \prod_{i=1}^{p} s_5(d_{k+1+i}) \ \Big|\ n \geq 1\Big\}.\end{aligned}$$

Let $C_7 = \{X_n^7 : n \geq 1\}$. The set of exits is $U(C_7) = \{X_0^7, Z_7\}$. Since C_7 is a conservative cluster for E_7 in I and since the set of exits is finite, we may apply $CFAR^\infty$. This yields for all $n \geq 1$, which we can immediately infer for X_0^7:

$$\tau_I(X_0^7) = \tau_I(X_n^7) = \tau \cdot \prod_{i=1}^{p} s_5(d_{k+1+i}).$$

Let $\beta_7 = \beta\big(I, \{c_1(d_{k+2}), c_3(\tilde{d}_{k+1})\}\big)$. We will show that, if we put for all $n \geq 0$

$$X_n^7 = \pi_J^p \circ \beta_7 \circ \partial_H\big(A_{k+1}(\tilde{d}_{k+1}^n)\big),$$

this will solve E_7. First let $n = 0$. We find easily

$$\begin{aligned}\pi_J^p \circ \beta_7 \circ \partial_H\big(A_{k+1}(\lambda)\big) &\overset{(38)}{=} c_1(d_{k+2}) \cdot \pi_J^p \circ \tau_I \circ \partial_H\big(B_{k+1}(\lambda)\big) \\ &\overset{(67)}{=} c_1(d_{k+2}) \cdot \tau \cdot \prod_{i=1}^{p} s_5(d_{k+1+i}).\end{aligned}$$

Now we will handle $n \geq 1$.

$$\begin{aligned}
\pi_J^p \circ \beta_7 \circ \partial_H\big(A_{k+1}(\tilde{d}_{k+1}^n)\big) &\overset{(39)}{=} l \cdot \pi_J^p \circ \beta_7 \circ \partial_H\big(A_{k+1}(\tilde{d}_{k+1}^{n-1})\big) \\
&\quad + c_1(d_{k+2}) \cdot \pi_J^p \circ \tau_I \circ \partial_H\big(B_{k+1}(\tilde{d}_{k+1}^{n-1})\big) \\
&\quad + c_3(\tilde{d}_{k+1}) \cdot \pi_J^p \circ \tau_I \circ \partial_H\big(I_k(\tilde{d}_{k+1}^{n-1})\big) \\
&\overset{(67)}{=} l \cdot \pi_J^p \circ \beta_7 \circ \partial_H\big(A_{k+1}(\tilde{d}_{k+1}^{n-1})\big) + X_0^7 \\
&\quad + c_3(\tilde{d}_{k+1}) \cdot \pi_J^p \circ \tau_I \circ \partial_H\big(I_k(\tilde{d}_{k+1}^{n-1})\big) \\
&\overset{(74)}{=} l \cdot \pi_J^p \circ \beta_7 \circ \partial_H\big(A_{k+1}(\tilde{d}_{k+1}^{n-1})\big) + X_0^7 + Z_7
\end{aligned}$$

Combining the just derived results we find with a commutativity/absorption argument that equation (76) is correct. Now we are in a position to prove that equation (66) is valid. We need yet another guarded recursive specification E_8 for that purpose.

$$\begin{aligned}
E_8 = \big\{\, X_0^8 &= t \cdot Y_0^8 + Z_8 \\
Y_0^8 &= c_2(\tilde{d}_{k+1}) \cdot X_1^8 \\
X_n^8 &= l \cdot X_{n-1}^8 + t \cdot Y_n^8 + Z_8 \\
Y_n^8 &= l \cdot Y_{n-1}^8 + c_2(\tilde{d}_{k+1}) \cdot X_{n+1}^8 \\
Z_8 &= c_4(\overline{k}) \cdot \prod_{i=1}^{p} s_5(d_{k+1+i}) \;\big|\; n \geq 1 \,\big\}.
\end{aligned}$$

Let $C_8 = \{X_n^8, Y_n^8 : n \geq 0\}$. The set of exits is $U(C_8) = \{Z_8\}$. It is clear that C_8 is conservative, so we apply $CFAR^\infty$ and we find for all $n \geq 0$

$$\tau_I(X_n^8) = \tau_I(Y_n^8) = \tau \cdot \prod_{i=1}^{p} s_5(d_{k+1+i}).$$

We will not need a new β. We will take $\beta_3 = \beta\big(I, \{c_4(\overline{k})\}\big)$. Now it will not be difficult to show that, if we put for all $n \geq 0$

$$\begin{aligned}
X_n^8 &= \pi_J^p \circ \beta_3 \circ \partial_H\big(F_k(\tilde{d}_{k+1}^n)\big) \\
Y_n^8 &= \pi_J^p \circ \beta_3 \circ \partial_H\big(H_k(\tilde{d}_{k+1}^n)\big),
\end{aligned}$$

this will solve E_8. Use equations (50), (51), (55) and (56) together with the just inferred result on the A-nodes (76). Combining the results with an easy commutativity/absorption argument gives that equation (66) is correct if we assume that equations (62)–(65) are valid.

Now we will prove that equation (66) itself is correct. We will do this with induction on p. So let $p = 1$. We will show that the assumptions that we made in equations (62)–(65) are valid for $p = 1$. Let us first take a look at the first assumption (62) and consider the following simple calculation.

$$\pi_J^1 \circ \beta_1 \circ \partial_H\big(D_k(\lambda)\big) \stackrel{(46)}{=} t \cdot \pi_J^1 \circ \beta_1 \circ \partial_H\big(E_k(\lambda)\big) + s_5(d_{k+1}).$$

So we see that this is just the assumption (62) for $p = 1$. Now let us verify that the second assumption (63) is also valid for $p = 1$.

$$\begin{aligned}\pi_J^1 \circ \beta_1 \circ \partial_H\big(D_k(\tilde{d}_{k+1}^n)\big) \stackrel{(47)}{=} & \; l \cdot \pi_J^1 \circ \beta_1 \circ \partial_H\big(D_k(\tilde{d}_{k+1}^{n-1})\big) \\ & + t \cdot \pi_J^1 \circ \beta_1 \circ \partial_H\big(E_k(\tilde{d}_{k+1}^n)\big) + s_5(d_{k+1}).\end{aligned}$$

This is precisely equation (63) for $p = 1$. The verification that the other two assumptions are correct for $p = 1$ can be proved analogously. So we obtain that equation (66) is correct for $p = 1$. This concludes the basis of our induction on p. Now assume that equation (66) is valid for one $p \geq 1$. We will show that the assumptions that we made are now valid for $p + 1$. First, we will verify that equation (62) is correct for $p + 1$. Consider thereto the following.

$$\begin{aligned}\pi_J^{p+1} \circ \beta_1 \circ \partial_H\big(D_k(\lambda)\big) \stackrel{(46)}{=} & \; t \cdot \pi_J^{p+1} \circ \beta_1 \circ \partial_H\big(E_k(\lambda)\big) \\ & + s_5(d_{k+1}) \cdot \pi_J^p \circ \tau_I \circ \partial_H\big(F_k(\lambda)\big) \\ = & \; t \cdot \pi_J^{p+1} \circ \beta_1 \circ \partial_H\big(E_k(\lambda)\big) \\ & + s_5(d_{k+1}) \cdot \tau \cdot \prod_{i=1}^{p} s_5(d_{k+1+i}) \\ = & \; t \cdot \pi_J^{p+1} \circ \beta_1 \circ \partial_H\big(E_k(\lambda)\big) \\ & + \prod_{i=1}^{p+1} s_5(d_{k+i}).\end{aligned}$$

We see that this is indeed equation (62) for $p + 1$. Now let us make a similar calculation for the second assumption.

$$\begin{aligned}\pi_J^{p+1} \circ \beta_1 \circ \partial_H\big(\big(D_k(\tilde{d}_{k+1}^n)\big) \stackrel{(47)}{=} & \; l \cdot \pi_J^{p+1} \circ \beta_1 \circ \partial_H\big(D_k(\tilde{d}_{k+1}^{n-1})\big) \\ & + t \cdot \pi_J^{p+1} \circ \beta_1 \circ \partial_H\big(E_k(\tilde{d}_{k+1}^n)\big) \\ & + s_5(d_{k+1}) \cdot \pi_J^p \circ \tau_I \circ \partial_H\big(F_k(\tilde{d}_{k+1}^n)\big) \\ = & \; l \cdot \pi_J^{p+1} \circ \beta_1 \circ \partial_H\big(D_k(\tilde{d}_{k+1}^{n-1})\big) \\ & + t \cdot \pi_J^{p+1} \circ \beta_1 \circ \partial_H\big(E_k(\tilde{d}_{k+1}^n)\big) \\ & + s_5(d_{k+1}) \cdot \tau \cdot \prod_{i=1}^{p} s_5(d_{k+1+i})\end{aligned}$$

$$\begin{aligned}&= l \cdot \pi_J^{p+1} \circ \beta_1 \circ \partial_H\big(D_k(\tilde{d}_{k+1}^{n-1})\big)\\&\quad + t \cdot \pi_J^{p+1} \circ \beta_1 \circ \partial_H\big(E_k(\tilde{d}_{k+1}^{n})\big)\\&\quad + \prod_{i=1}^{p+1} s_5(d_{k+i}).\end{aligned}$$

We see that this is exactly the second assumption (63) for $p+1$. The proof that the assumptions (64) and (65) are also correct for $p+1$ can be calculated in the same way. But then we can immediately conclude that (66) is correct for $p+1$. This ends our induction on p and we find that equation (66) is correct for all p. With this knowledge it is evident that the assumptions that we made are also correct by themselves. (For, the case $p = 1$ was already correct and for $p > 1$ we can reiterate, using (66), the above calculation.) We will use this fact to prove that the following holds for all $n \geq 0$.

$$\left.\begin{aligned}\pi_J^p \circ \tau_I \circ \partial_H\big(D_k(\tilde{d}_{k+1}^{n})\big)\\ \pi_J^p \circ \tau_I \circ \partial_H\big(E_k(\tilde{d}_{k+1}^{n})\big)\end{aligned}\right\} = \tau \cdot \prod_{i=1}^{p} s_5(d_{k+i}). \tag{77}$$

Consider the subsequent guarded recursive specification E_9.

$$\begin{aligned}E_9 = \big\{\, X_0^9 &= t \cdot Y_0^9 + Z_9\\ Y_0^9 &= c_2(\tilde{d}_{k+1}) \cdot X_1^9 + Z_9\\ X_n^9 &= l \cdot X_{n-1}^9 + t \cdot Y_n^9 + Z_9\\ Y_n^9 &= l \cdot Y_{n-1}^9 + c_2(\tilde{d}_{k+1}) \cdot X_{n+1}^9 + Z_9\\ Z_9 &= \prod_{i=1}^{p} s_5(d_{k+i}) \;\big|\; n \geq 1 \,\big\}.\end{aligned}$$

We see at once from equations (62)–(65) that if we put for all $n \geq 0$

$$\begin{aligned}X_n^9 &= \pi_J^p \circ \beta_1 \circ \partial_H\big(D_k(\tilde{d}_{k+1}^{n})\big)\\ Y_n^9 &= \pi_J^p \circ \beta_1 \circ \partial_H\big(E_k(\tilde{d}_{k+1}^{n})\big),\end{aligned}$$

that this will solve the system E_9. Let $C_9 = \{X_n^9, Y_n^9 : n \geq 0\}$. The set of exits is $U(C_9) = \{Z_9\}$. We may apply $CFAR^\infty$ and we find for all $n \geq 0$

$$\tau_I(X_n^9) = \tau_I(Y_n^9) = \tau \cdot \prod_{i=1}^{p} s_5(d_{k+i}).$$

A simple commutativity/absorption argument will yield that equation (77) is valid. In the sequel we will need the following result for a single *BC*-trace. For all $n \geq 0$

$$\left.\begin{aligned}\pi_J^p \circ \tau_I \circ \partial_H\big(B_k(\tilde{d}_{k+1}^{n})\big)\\ \pi_J^p \circ \tau_I \circ \partial_H\big(C_k(\tilde{d}_{k+1}^{n})\big)\end{aligned}\right\} = \tau \cdot \prod_{i=1}^{p} s_5(d_{k+i}). \tag{78}$$

To achieve this, we will need one more guarded recursive specification E_{10}.

$$\begin{aligned}
E_{10} = \{\ X_0^{10} &= c_2(\tilde{d}_{k+1}) \cdot Y_1^{10} \\
Y_0^{10} &= t \cdot X_0^{10} \\
X_n^{10} &= l \cdot X_{n-1}^{10} + c_2(\tilde{d}_{k+1}) \cdot Y_{n+1}^{10} + Z_{10} \\
Y_n^{10} &= l \cdot Y_{n-1}^{10} + t \cdot X_n^{10} + Z_{10} \\
Z_{10} &= c_3(\tilde{d}_{k+1}) \cdot \prod_{i=1}^{p} s_5(d_{k+i}) \bigm| n \geq 1 \,\}.
\end{aligned}$$

Let $C_{10} = \{X_n^{10}, Y_n^{10} : n \geq 0\}$. It will be clear that this is a conservative cluster for E_{10} in I. The set of exits is $U(C_{10}) = \{Z_{10}\}$. So if we apply *CFAR*$^\infty$ we find for all $n \geq 0$

$$\tau_I(X_n^{10}) = \tau_I(Y_n^{10}) = \tau \cdot \prod_{i=1}^{p} s_5(d_{k+i}).$$

Let $\beta_8 = \beta\big(I, \{c_3(\tilde{d}_{k+1})\}\big)$. With the aid of equations (40), (41), (43) and (44) together with the just derived result on the *DE*-traces (77) it is not hard to see that if we put for all $n \geq 0$

$$\begin{aligned}
X_n^{10} &= \pi_J^p \circ \beta_8 \circ \partial_H\big(B_k(\tilde{d}_{k+1}^n)\big) \\
Y_n^{10} &= \pi_J^p \circ \beta_8 \circ \partial_H\big(C_k(\tilde{d}_{k+1}^n)\big),
\end{aligned}$$

this is a solution for the guarded recursive specification E_{10}. So a straightforward calculation gives us that equation (78) is correct.

Hereinafter, we will give the final calculation in order to prove 4.4.14. Choose $k = 0$ and consider the following.

$$\begin{aligned}
\pi_J^p \circ \tau_I \circ \partial_H\big(A_0(\lambda)\big) &\overset{(38)}{=} \tau \cdot \pi_J^p \circ \tau_I \circ \partial_H\big(B_0(\lambda)\big) \\
&\overset{(78)}{=} \tau \cdot \prod_{i=1}^{p} s_5(d_i) \\
&= \tau \cdot \pi_J^p\Big(\prod_{i=1}^{\infty} s_5(d_i)\Big).
\end{aligned}$$

So we see that we have shown for a fixed $p \geq 1$

$$\pi_J^p \circ \tau_I \circ \partial_H\Big(S(\overline{0}) \parallel Q(\lambda) \parallel R(\overline{0}) \parallel \prod_{i=1}^{\infty} s_1(d_i)\Big) = \tau \cdot \pi_J^p\Big(\prod_{i=1}^{\infty} s_5(d_i)\Big).$$

With the use of *GIP* we conclude the proof of 4.4.14.

In the following application we will increase the unreliability of the queue. We will allow that the queue can lose or duplicate any datum contained in it at any moment. In order to be able to give a guarded recursive specification for this faulty and stuttering queue, we will need the notion of a residue set and a stuttering set of the words that the faulty and stuttering queue can contain. The notion of a residue set is taken from [**22**].

Definitions (4.4.15)

Let $n \geq 1$ and let $\sigma \in E^*$ be a non-empty word, say $\sigma = e_1 * e_2 * \cdots * e_n$. For $n \geq 2$ the residue set $R(\sigma)$ of σ is the following set

$$R(\sigma) = \{e_1 * \cdots * e_{n-1}, e_2 * \cdots * e_n\} \cup \{e_1 * \cdots * e_{i-1} * e_{i+1} * \cdots * e_n : 1 < i < n\}$$

and if $n = 1$ we will have for the residue set $R(e_1) = \{\lambda\}$. For the stuttering set we will not need this dichotomy. The stuttering set $S(\sigma)$ of σ is the following set

$$S(\sigma) = \{e_1 * \cdots * e_i * e_i * \cdots * e_n : 1 \leq i \leq n\}.$$

Just as with the previous queue we will express the lossy behaviour by an atomic action l. The stuttering will be expressed with an atomic action s. Now we are in a position to give the guarded recursive specification for the faulty and stuttering queue.

$$\begin{aligned} Q(\lambda) &= \sum_{x \in E} r_2(x) \cdot Q(x) \\ Q(x * \sigma) &= \sum_{y \in E} r_2(y) \cdot Q(x * \sigma * y) + s_3(x) \cdot Q(\sigma) \\ &\qquad + \sum_{\rho_1 \in R(x*\sigma)} l \cdot Q(\rho_1) + \sum_{\rho_2 \in S(x*\sigma)} s \cdot Q(\rho_2). \end{aligned}$$

The other components of our alternating bit protocol will remain the same just as in the former theorem. We will reiterate the encapsulation set H and we will give a new abstraction set I that only differs from the previous one in having an extra s.

$$\begin{aligned} H = &\left\{ r_1(d), s_1(d) : d \in D \right\} \\ &\cup \left\{ r_2(x), r_3(x), s_2(x), s_3(x) : x \in E \right\} \\ &\cup \left\{ r_4(\overline{k}), s_4(\overline{k}) : \overline{k} \in \mathbb{Z}/2\mathbb{Z} \right\} \\ I = &\left\{ c_1(d) : d \in D \right\} \\ &\cup \left\{ c_2(x), c_3(x) : x \in E \right\} \\ &\cup \left\{ c_4(\overline{k}) : \overline{k} \in \mathbb{Z}/2\mathbb{Z} \right\} \cup \{l, t, s\}. \end{aligned}$$

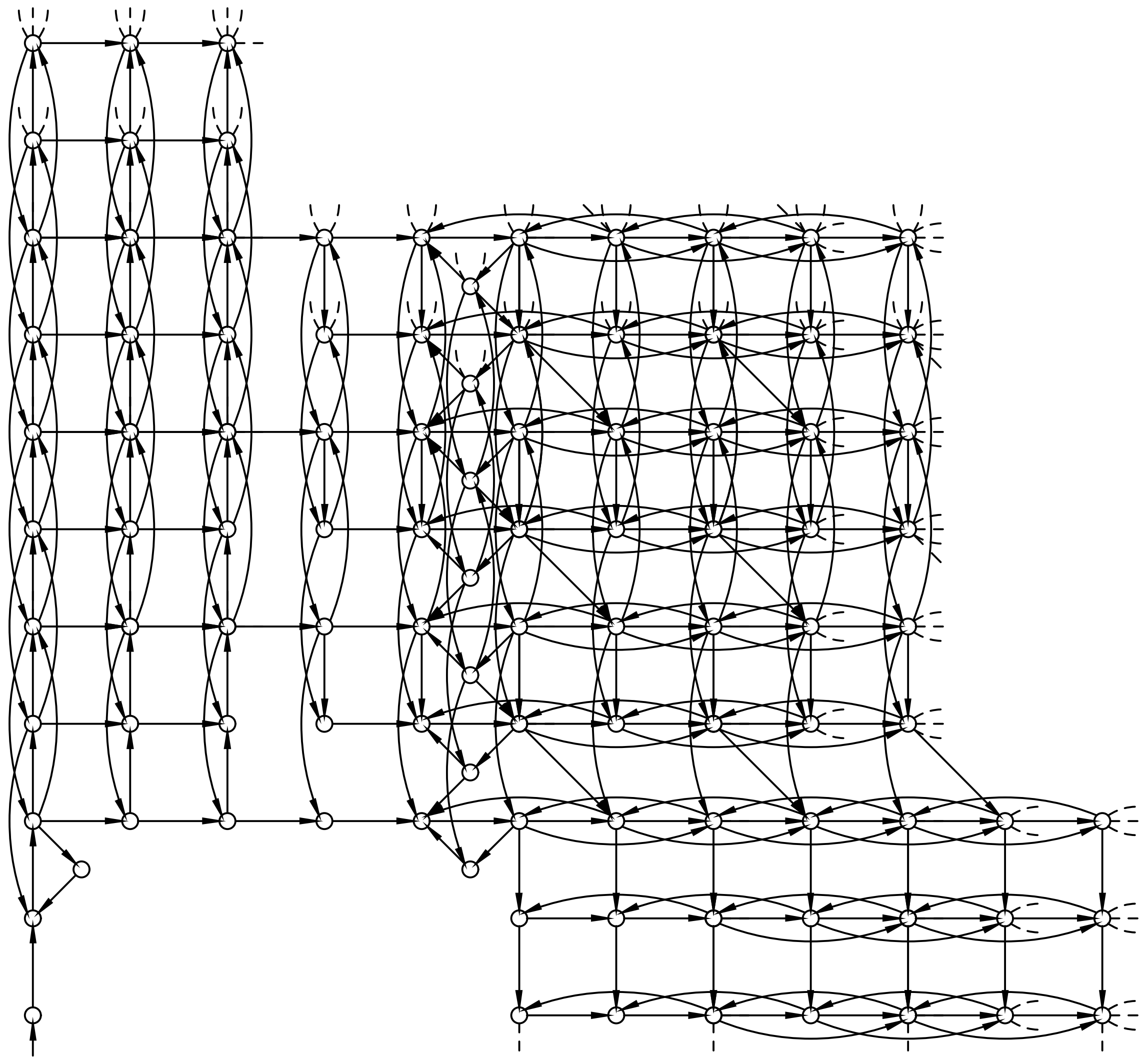

Figure 4.10. Another state transition diagram.

Theorem (4.4.16)

Let Q be the faulty and stuttering queue and let the other components be as in theorem (4.4.8). Then the following holds for all $(d_i)_i \in D^\omega$

$$\tau_I \circ \partial_H\Big(S(\overline{0}) \parallel Q(\lambda) \parallel R(\overline{0}) \parallel \prod_{i=1}^{\infty} s_1(d_i)\Big) = \tau \cdot \prod_{i=1}^{\infty} s_5(d_i).$$

Proof. The structure of this proof will not be entirely the same as before. Let $(d_i)_i \in D^\omega$ be fixed. We will first calculate

$$\partial_H\Big(S(\overline{0}) \parallel Q(\lambda) \parallel R(\overline{0}) \parallel \prod_{i=1}^{\infty} s_1(d_i)\Big).$$

We have schematically depicted this process in figure 4.10 on page 217. We will leave out the actual calculation of this process but we will use figure 4.10 to provide some explanation on the resulting guarded recursive specification. For the guarded recursive specification that follows we will use the abbreviations

$$A_k(\sigma), B_k(\sigma), \ldots, J_k(\sigma)$$

that we already saw in theorems (4.4.8) and (4.4.14). We will alternate pieces of the guarded recursive specification with informal comment on them to optimize the readability of the specification. Moreover, the order of the equations will be different than in the former theorems. It is determined by a walk through figure 4.10.

We suppose that k, n, m range over the set of natural numbers with $k \geq 0$ and $n, m \geq 1$. Informally, we will start walking at the initial point $\partial_H\big(A_0(\lambda)\big)$ of the graph in figure 4.10. Formally, we will start for all $k \geq 0$ with equation $\partial_H\big(A_k(\lambda)\big)$. The first equation corresponds to the transition from the initial node to the only node that can be reached from that point: the second node. This equation is the same as it appeared in theorem (4.4.14).

$$\partial_H\big(A_k(\lambda)\big) = c_1(d_{k+1}) \cdot \partial_H\big(B_k(\lambda)\big).$$

This is not surprising since in this phase the faulty and stuttering queue is empty. In the second node the queue is still empty so this equation will not differ from the one in the former theorem either.

$$\partial_H\big(B_k(\lambda)\big) = c_2(\tilde{d}_{k+1}) \cdot \partial_H\big(C_k(\tilde{d}_{k+1})\big).$$

In the third node several things can occur: a time out can be given (we will go up one node in the graph), a loss of any datum can be given (we will go southeastwards and enter the triangle), a duplication of any datum can be performed (we will rise two nodes in the graph with an arched edge) or a communication can be executed (we will go to the right in the graph). This behaviour will be repeated in the node that we enter if the duplication has taken place and so on, see the picture. So we obtain for all these nodes the following equation

$$\begin{aligned}\partial_H\big(C_k(\tilde{d}^n_{k+1})\big) = l \cdot \partial_H\big(C_k(\tilde{d}^{n-1}_{k+1})\big) + s \cdot \partial_H\big(C_k(\tilde{d}^{n+1}_{k+1})\big) \\ + t \cdot \partial_H\big(B_k(\tilde{d}^n_{k+1})\big) + c_3(\tilde{d}_{k+1}) \cdot \partial_H\big(D_k(\tilde{d}^{n-1}_{k+1})\big).\end{aligned}$$

If $n = 1$ we will enter the triangle. In that node only a time out can be performed and we come back in the second node. This corresponds to the equation

$$\partial_H\big(C_k(\lambda)\big) = t \cdot \partial_H\big(B_k(\lambda)\big).$$

Half of the first trace up is now clarified but we must take a look at the even nodes of this trace. As with the odd nodes four actions can be executed. We can lose any datum element and go down two nodes with an arched edge, a duplication of any datum can be performed and we will go up two nodes with an arched edge, the queue can be loaded with one more datum of the same type (we will go up one node in the graph) or a communication can take place and a datum will be removed from the queue (we will go to the right in the graph to the second trace up). Observe that we are dealing with a queue that contains one type of data, so the residue set and the stuttering set both contain one element so there is one stuttering edge and one losing edge. We will have the following equation for these even nodes

$$\begin{aligned}\partial_H\bigl(B_k(\tilde{d}^n_{k+1})\bigr) &= l\cdot\partial_H\bigl(B_k(\tilde{d}^{n-1}_{k+1})\bigr) + s\cdot\partial_H\bigl(B_k(\tilde{d}^{n+1}_{k+1})\bigr)\\ &\qquad + c_2(\tilde{d}_{k+1})\cdot\partial_H\bigl(C_k(\tilde{d}^{n+1}_{k+1})\bigr) + c_3(\tilde{d}_{k+1})\cdot\partial_H\bigl(E_k(\tilde{d}^{n-1}_{k+1})\bigr).\end{aligned}$$

This trace up will be called a single BC-trace since the queue contains only a single type of elements. We have seen that the only way out from a single BC-trace is by entering what we will call a DE-trace. We will discuss this second trace. We will start with the lowest node of the second trace. In this phase the queue is empty. A time out can be executed and we will go up one node or a send action at port 5 can take place (we go to the right in the graph). This corresponds to the following equation

$$\partial_H\bigl(D_k(\lambda)\bigr) = t\cdot\partial_H\bigl(E_k(\lambda)\bigr) + s_5(d_{k+1})\cdot\partial_H\bigl(F_k(\lambda)\bigr).$$

Now we will discuss the second node of the DE-trace. At that node we can load the queue with a datum element or we can perform the send action to the right, as well. So we obtain a similar equation.

$$\partial_H\bigl(E_k(\lambda)\bigr) = c_2(\tilde{d}_{k+1})\cdot\partial_H\bigl(D_k(\tilde{d}_{k+1})\bigr) + s_5(d_{k+1})\cdot\partial_H\bigl(H_k(\lambda)\bigr).$$

Now we take a node of this trace that is neither the first nor the second. Then we will have two more possible actions that can be performed since the queue it not empty at this stage: an additional duplication or deletion can be carried out. We can see this in the graph. The following two equations will complete the description of the DE-trace.

$$\begin{aligned}\partial_H\bigl(D_k(\tilde{d}^n_{k+1})\bigr) &= l\cdot\partial_H\bigl(D_k(\tilde{d}^{n-1}_{k+1})\bigr) + s\cdot\partial_H\bigl(D_k(\tilde{d}^{n+1}_{k+1})\bigr)\\ &\qquad + t\cdot\partial_H\bigl(E_k(\tilde{d}^{n}_{k+1})\bigr) + s_5(d_{k+1})\cdot\partial_H\bigl(F_k(\tilde{d}^{n}_{k+1})\bigr)\\ \partial_H\bigl(E_k(\tilde{d}^n_{k+1})\bigr) &= l\cdot\partial_H\bigl(E_k(\tilde{d}^{n-1}_{k+1})\bigr) + s\cdot\partial_H\bigl(E_k(\tilde{d}^{n+1}_{k+1})\bigr)\\ &\qquad + c_2(\tilde{d}_{k+1})\cdot\partial_H\bigl(D_k(\tilde{d}^{n+1}_{k+1})\bigr) + s_5(d_{k+1})\cdot\partial_H\bigl(H_k(\tilde{d}^{n}_{k+1})\bigr).\end{aligned}$$

As with the single BC-trace the way out from the DE-trace is by performing a send action and entering an FH-trace. We will start at the lowest level of this trace. The queue is empty and just two actions can be done: a time out can be given and we will go up in the *FH*-trace or a communication can take place that acknowledges the transmission of the send action that was executed before (when leaving the *DE*-trace) and we will enter the subsequent A-node (we will go to the right). This corresponds to the following equation

$$\partial_H\big(F_k(\lambda)\big) = t \cdot \partial_H\big(H_k(\lambda)\big) + c_4(\overline{k}) \cdot \partial_H\big(A_{k+1}(\lambda)\big).$$

Now we will discuss the second node of the *FH*-trace. In that phase the queue is still empty, for, we entered it due to a time out. There is just one action that can be executed: the queue will be loaded with a datum element (the one that was just sent away). This will be represented by the following equation

$$\partial_H\big(H_k(\lambda)\big) = c_2(\tilde{d}_{k+1}) \cdot \partial_H\big(F_k(\tilde{d}_{k+1})\big).$$

Taking a node of the *FH*-trace that is neither the first nor the second will yield two more possibilities: duplication or deletion of any datum. We can see this in the picture. Only in the odd nodes we can enter a dense block with entrances that are A-nodes. See the following two equations that complete the discussion of the *FH*-traces.

$$\begin{aligned}\partial_H\big(F_k(\tilde{d}^n_{k+1})\big) &= l \cdot \partial_H\big(F_k(\tilde{d}^{n-1}_{k+1})\big) + s \cdot \partial_H\big(F_k(\tilde{d}^{n+1}_{k+1})\big)\\ &\quad + t \cdot \partial_H\big(H_k(\tilde{d}^n_{k+1})\big) + c_4(\overline{k}) \cdot \partial_H\big(A_{k+1}(\tilde{d}^n_{k+1})\big)\\ \partial_H\big(H_k(\tilde{d}^n_{k+1})\big) &= l \cdot \partial_H\big(H_k(\tilde{d}^{n-1}_{k+1})\big) + s \cdot \partial_H\big(H_k(\tilde{d}^{n+1}_{k+1})\big)\\ &\quad + c_2(\tilde{d}_{k+1}) \cdot \partial_H\big(F_k(\tilde{d}^{n+1}_{k+1})\big).\end{aligned}$$

The only nodes that we did not discuss yet are the A-nodes that contain datum elements that are old: the send action has been executed but not the acknowledgement and due to time outs the queue will be loaded with old data. This can be clearly seen in the description of the *FH*-trace. So we will discuss here the A-nodes with a non-empty queue. In this phase duplication and deletion can take place (these correspond in figure 4.10 to the up and down going arched edges). Besides this behaviour a new datum element can be taken from the test row (we go to the right in the picture and we enter a B-node) or a communication can take place with the receiver such that one datum can be removed from the queue (we will go down one node and enter a I-node). The equation that accompanies this is as follows.

$$\begin{aligned}\partial_H\big(A_{k+1}(\tilde{d}^n_{k+1})\big) &= l \cdot \partial_H\big(A_{k+1}(\tilde{d}^{n-1}_{k+1})\big)\\ &\quad + s \cdot \partial_H\big(A_{k+1}(\tilde{d}^{n+1}_{k+1})\big)\\ &\quad + c_1(d_{k+2}) \cdot \partial_H\big(B_{k+1}(\tilde{d}^n_{k+1})\big)\\ &\quad + c_3(\tilde{d}_{k+1}) \cdot \partial_H\big(I_k(\tilde{d}^{n-1}_{k+1})\big).\end{aligned}$$

First, we will treat the I-nodes. From the lowest one we can do just one thing: take the following element of the test row (we go to the right in the picture). On the other levels the queue is not empty so we will also have the possibility to duplicate or to delete (the arched up and down going edges in the figure). We will give both equations for the I-nodes.

$$\begin{aligned}\partial_H\bigl(I_k(\lambda)\bigr) &= c_1(d_{k+2})\cdot\partial_H\bigl(J_k(\lambda)\bigr)\\ \partial_H\bigl(I_k(\tilde{d}^n_{k+1})\bigr) &= l\cdot\partial_H\bigl(I_k(\tilde{d}^{n-1}_{k+1})\bigr)+s\cdot\partial_H\bigl(I_k(\tilde{d}^{n+1}_{k+1})\bigr)\\ &\quad+c_1(d_{k+2})\cdot\partial_H\bigl(J_k(\tilde{d}^n_{k+1})\bigr).\end{aligned}$$

Now we will treat the B-nodes. Recall that they are at the right of the A-nodes. We can lose or duplicate or we can load or communicate. After deletions and stutterings we will still be in a B-node (the arched edges). If we load we will go to the right in the picture (we will enter the double BC-trace since there are now two types of data) and if we communicate we will go one node down in the graph. We will give the equation of the B-nodes.

$$\begin{aligned}\partial_H\bigl(B_{k+1}(\tilde{d}^n_{k+1})\bigr) &= l\cdot\partial_H\bigl(B_{k+1}(\tilde{d}^{n-1}_{k+1})\bigr)+s\cdot\partial_H\bigl(B_{k+1}(\tilde{d}^{n+1}_{k+1})\bigr)\\ &\quad+c_2(\tilde{d}_{k+2})\cdot\partial_H\bigl(C_{k+1}(\tilde{d}^n_{k+2}*\tilde{d}_{k+2})\bigr)\\ &\quad+c_3(\tilde{d}_{k+1})\cdot\partial_H\bigl(J_k(\tilde{d}^{n-1}_{k+1})\bigr).\end{aligned}$$

Observe that we have in the node that is right from this B-node two types of data in the queue: old data and a new one. Since we can lose or duplicate any datum that is in the queue at any moment we will see in the graph more edges than in the first part of the figure: we have two deletions and two duplications. The triangles that we can see in the picture are due to the fact that the just read new datum can be deleted immediately. This means that there are also C-nodes. First, we will discuss the node with the old and new data. As we already said there are two types of deletion and duplication (arched edges to the right and two nodes up are the stuttering edges; an arched edge two nodes down and the small edge to the triangle node are the deletion edges). Also a time out can be given (we will go to the right one node in the graph). Finally, a communication with the receiver can be executed and we will go down one node in the graph. This is expressed in the following equation (recall that $n, m \geq 1$).

$$\begin{aligned}\partial_H\bigl(C_{k+1}(\tilde{d}^n_{k+1}*\tilde{d}^m_{k+2})\bigr) &= l\cdot\partial_H\bigl(C_{k+1}(\tilde{d}^{n-1}_{k+1}*\tilde{d}^m_{k+2})\bigr)\\ &\quad+l\cdot\partial_H\bigl(C_{k+1}(\tilde{d}^n_{k+1}*\tilde{d}^{m-1}_{k+2})\bigr)\\ &\quad+s\cdot\partial_H\bigl(C_{k+1}(\tilde{d}^{n+1}_{k+1}*\tilde{d}^m_{k+2})\bigr)\\ &\quad+s\cdot\partial_H\bigl(C_{k+1}(\tilde{d}^n_{k+1}*\tilde{d}^{m+1}_{k+2})\bigr)\\ &\quad+t\cdot\partial_H\bigl(B_{k+1}(\tilde{d}^n_{k+1}*\tilde{d}^m_{k+2})\bigr)\\ &\quad+c_3(\tilde{d}_{k+1})\cdot\partial_H\bigl(G_k(\tilde{d}^{n-1}_{k+1}*\tilde{d}^m_{k+2})\bigr).\end{aligned}$$

Now we will discuss the C-nodes. In these nodes there is only one type of data left: the old ones. So we will have here one type of deletion and duplication (the arched edges up and down at the triangle nodes). A time out can be performed (we will go northwestwards in the triangle and return to the B-node) or we can communicate with the receiver and an old datum can be removed from the queue (we will go southeastwards in the graph). We will give the corresponding equation in the next display.

$$\begin{aligned}\partial_H\big(C_{k+1}(\tilde{d}^{n}_{k+1})\big) = {} & l\cdot\partial_H\big(C_{k+1}(\tilde{d}^{n-1}_{k+1})\big) + s\cdot\partial_H\big(C_{k+1}(\tilde{d}^{n+1}_{k+1})\big)\\ & + t\cdot\partial_H\big(B_{k+1}(\tilde{d}^{n}_{k+1})\big) + c_3(\tilde{d}_{k+1})\cdot\partial_H\big(G_k(\tilde{d}^{n-1}_{k+1})\big).\end{aligned}$$

We will finish the discussion of the double BC-trace by giving the equation for the B-nodes that contain two types of data. Just as with the C-nodes of this trace we will have two types of deletion and duplication (for the stuttering we go up and to the right with an arched edge and for the deletion we will go down and to the left with an arched edge). Besides these things we can also stay in the double BC-trace by loading one more new datum in the queue (we go one node to the right) or we can escape it by communicating with the receiver (we go one node down in the picture). We will give the equation for all these nodes.

$$\begin{aligned}\partial_H\big(B_{k+1}(\tilde{d}^{n}_{k+1} * \tilde{d}^{m}_{k+2})\big) = {} & l\cdot\partial_H\big(B_{k+1}(\tilde{d}^{n-1}_{k+1} * \tilde{d}^{m}_{k+2})\big)\\ & + l\cdot\partial_H\big(B_{k+1}(\tilde{d}^{n}_{k+1} * \tilde{d}^{m-1}_{k+2})\big)\\ & + s\cdot\partial_H\big(B_{k+1}(\tilde{d}^{n+1}_{k+1} * \tilde{d}^{m}_{k+2})\big)\\ & + s\cdot\partial_H\big(B_{k+1}(\tilde{d}^{n}_{k+1} * \tilde{d}^{m+1}_{k+2})\big)\\ & + c_2(\tilde{d}_{k+2})\cdot\partial_H\big(C_{k+1}(\tilde{d}^{n}_{k+1} * \tilde{d}^{m+1}_{k+2})\big)\\ & + c_3(\tilde{d}_{k+1})\cdot\partial_H\big(J_k(\tilde{d}^{n-1}_{k+1} * \tilde{d}^{m}_{k+2})\big).\end{aligned}$$

To escape a single BC-trace we communicate with the receiver and we enter a DE-trace. With the double BC-trace we communicate with the receiver and we will enter the JG-trace. These traces are the only ones that remain for our discussion. For both nodes we will have four equations. First, we will treat the nodes with an empty queue. The location of the empty G-node in the picture is the lowest node of the dense block; it is part of a triangle. If we go from that point northwestwardly to the next node we will enter the empty J-node. In this node the only possible action is the loading of the queue with a new datum element (we will go to the right in the picture). This is expressed in an equation as follows.

$$\partial_H\big(J_k(\lambda)\big) = c_2(\tilde{d}_{k+2})\cdot\partial_H\big(G_k(\tilde{d}_{k+2})\big).$$

We see that in this equation there is a G-node with only a new datum. We will discuss this node after the empty G-node. In this node we have two possible actions. We can perform a time out (we go northwestwardly to the empty J-node) or an acknowledgement can be executed and we enter the beginning of the next single BC-trace. This is the corresponding equation

$$\partial_H\big(G_k(\lambda)\big) = t \cdot \partial_H\big(J_k(\lambda)\big) + c_4(\overline{k}) \cdot \partial_H\big(B_{k+1}(\lambda)\big).$$

The lowest horizontal line of the dense block is a somewhat different JG-trace than the others: in this trace all the old datum elements are removed from the queue and only new data is in the queue. Therefore, there is one type of deletion and duplication (we go to the left or to the right with arched edges). At a G-node we can execute a time out and we stay in the JG-trace (we go one node to the right in the figure) or we can perform an acknowledgement and we go to the next single BC-trace (we go to the southeast in the graph). For the J-nodes there is besides the stuttering and deletion actions just one possible action: the queue will be loaded with a new datum element (we will go one node to the right in the graph). We will give both equations hereinafter.

$$\begin{aligned}
\partial_H\big(G_k(\tilde{d}^m_{k+2})\big) &= l \cdot \partial_H\big(G_k(\tilde{d}^{m-1}_{k+2})\big) + s \cdot \partial_H\big(G_k(\tilde{d}^{m+1}_{k+2})\big) \\
&\quad + t \cdot \partial_H\big(J_k(\tilde{d}^m_{k+2})\big) + c_4(\overline{k}) \cdot \partial_H\big(B_{k+1}(\tilde{d}^m_{k+2})\big) \\
\partial_H\big(J_k(\tilde{d}^m_{k+2})\big) &= l \cdot \partial_H\big(J_k(\tilde{d}^{m-1}_{k+2})\big) + s \cdot \partial_H\big(J_k(\tilde{d}^{m+1}_{k+2})\big) \\
&\quad + c_2(\tilde{d}_{k+2}) \cdot \partial_H\big(G_k(\tilde{d}^{m+1}_{k+2})\big).
\end{aligned}$$

The JG-traces have at their beginning a triangle as with the beginning of the double BC-traces. In fact, we have in this dense block a horizontal CG-trace. The C-nodes of this trace have already been treated. Now we will handle the other part of this CG-trace. Since the empty G-node is already discussed we will treat the non-empty ones. In these nodes there are only old datum elements. Hence, we will have one deletion and one duplication (see the up and down going arched edges). A time out can be executed (we will go northwestwardly) and we return in the first J-node of the JG-trace or an acknowlegdement is performed (we will go southwestwards in the graph). We will give the equation.

$$\begin{aligned}
\partial_H\big(G_k(\tilde{d}^n_{k+1})\big) &= l \cdot \partial_H\big(G_k(\tilde{d}^{n-1}_{k+1})\big) + s \cdot \partial_H\big(G_k(\tilde{d}^{n+1}_{k+1})\big) \\
&\quad + t \cdot \partial_H\big(J_k(\tilde{d}^n_{k+1})\big) + c_4(\overline{k}) \cdot \partial_H\big(B_{k+1}(\tilde{d}^n_{k+1})\big).
\end{aligned}$$

We see in this equation that also J-nodes can contain only old data. We will discuss them hereinafter. Besides losing and stuttering (we will go up and down with arched edges) the queue can be loaded with a new datum element (we go one node to the right in the picture). We will give this equation below.

$$\begin{aligned}
\partial_H\big(J_k(\tilde{d}^n_{k+1})\big) &= l \cdot \partial_H\big(J_k(\tilde{d}^{n-1}_{k+1})\big) + s \cdot \partial_H\big(J_k(\tilde{d}^{n+1}_{k+1})\big) \\
&\quad + c_2(\tilde{d}_{k+2}) \cdot \partial_H\big(G_k(\tilde{d}^n_{k+1} * \tilde{d}_{k+2})\big).
\end{aligned}$$

The two remaining equations concern the JG-traces with two types of datum elements. For the G-nodes we have beside the four stuttering and deletion actions (see the four arched edges in the graph) two possible actions: we can stay in the JG-trace and perform a time out (we go one node to the right in the graph) or an acknowledgement is executed (we will go southeastwards to the lower double BC-trace). We will give its equation.

$$\begin{aligned}
\partial_H\big(G_k(\tilde{d}_{k+1}^n * \tilde{d}_{k+2}^m)\big) = {} & l \cdot \partial_H\big(G_k(\tilde{d}_{k+1}^{n-1} * \tilde{d}_{k+2}^m)\big) \\
& + l \cdot \partial_H\big(G_k(\tilde{d}_{k+1}^n * \tilde{d}_{k+2}^{m-1})\big) \\
& + s \cdot \partial_H\big(G_k(\tilde{d}_{k+1}^{n+1} * \tilde{d}_{k+2}^m)\big) \\
& + s \cdot \partial_H\big(G_k(\tilde{d}_{k+1}^n * \tilde{d}_{k+2}^{m+1})\big) \\
& + t \cdot \partial_H\big(J_k(\tilde{d}_{k+1}^n * \tilde{d}_{k+2}^m)\big) \\
& + c_4(\overline{k}) \cdot \partial_H\big(B_{k+1}(\tilde{d}_{k+1}^n * \tilde{d}_{k+2}^m)\big).
\end{aligned}$$

Finally, for the J-nodes with two types of data we have apart from the four obvious possibilities (see the arched edges in the graph) just one other action: the queue will be loaded with a new datum element (we go one node to the right in the picture). We will give the equation for these nodes.

$$\begin{aligned}
\partial_H\big(J_k(\tilde{d}_{k+1}^n * \tilde{d}_{k+2}^m)\big) = {} & l \cdot \partial_H\big(J_k(\tilde{d}_{k+1}^{n-1} * \tilde{d}_{k+2}^m)\big) \\
& + l \cdot \partial_H\big(J_k(\tilde{d}_{k+1}^n * \tilde{d}_{k+2}^{m-1})\big) \\
& + s \cdot \partial_H\big(J_k(\tilde{d}_{k+1}^{n+1} * \tilde{d}_{k+2}^m)\big) \\
& + s \cdot \partial_H\big(J_k(\tilde{d}_{k+1}^n * \tilde{d}_{k+2}^{m+1})\big) \\
& + c_2(\tilde{d}_{k+2}) \cdot \partial_H\big(G_k(\tilde{d}_{k+1}^n * \tilde{d}_{k+2}^{m+1})\big).
\end{aligned}$$

In the former theorems we used the large guarded recursive specification in the sequel of the proof. If one tries to imitate the proofs that have been exposed thus far, it will turn out that there is a problem with the dense block of figure 4.10: in the former verifications these blocks have been teared down with induction on the "floor height". We can see in figure 4.9 on page 201 that there are no arrows that go up from a floor; they all go to the right, down or to the southeast. In the dense block of figure 4.10 there are arrows that go up from all but the lowest floor. This is due to the duplication of old data. If we assume that only the last read datum element can be duplicated then we can use the above deduced guarded recursive specification. But now we can not use this induction argument since we can reach all the floors above us. What we will use is the fact that this dense block not just looks like a cluster: it can be made into a cluster in the sense of definition (4.4.10). To achieve the desired situation we have to introduce a new guarded recursive specification that is heavily based on the one that we found above. We will write down an acuted version of the above guarded recursive specification in which we will also acute some of the atomic actions that appear in the dense block. All the atomic actions in the dense block that result in a transition such that the queue becomes empty or does not contain old datum elements anymore will be acuted. Before we will explain more of the structure of this proof we will enumerate the guarded recursive specification and thereinafter we will show that we may use this specification in order to prove 4.4.16. Let k, n, m range over the set of natural numbers with $k \geq 0$, $n, m \geq 1$. In the subsequent display we will asume that $k \geq 0$ and $m \geq 1$ in all the equations. For $n \geq 1$ we will mention explicitly what the range is.

$$A'_k(\lambda) = c_1(d_{k+1}) \cdot B'_k(\lambda) \tag{79}$$

$$\begin{aligned} A'_{k+1}(\tilde{d}^n_{k+1}) = {} & l \cdot A'_{k+1}(\tilde{d}^{n-1}_{k+1}) + s \cdot A'_{k+1}(\tilde{d}^{n+1}_{k+1}) && (n \geq 1) \\ & + c_1(d_{k+2}) \cdot B'_{k+1}(\tilde{d}^n_{k+1}) + c_3(\tilde{d}_{k+1}) \cdot I'_k(\tilde{d}^{n-1}_{k+1}) && (80) \end{aligned}$$

$$B'_k(\lambda) = c_2(\tilde{d}_{k+1}) \cdot C'_k(\tilde{d}_{k+1}) \tag{81}$$

$$\begin{aligned} B'_k(\tilde{d}^n_{k+1}) = {} & l \cdot B'_k(\tilde{d}^{n-1}_{k+1}) + s \cdot B'_k(\tilde{d}^{n+1}_{k+1}) && (n \geq 1) \\ & + c_2(\tilde{d}_{k+1}) \cdot C'_k(\tilde{d}^{n+1}_{k+1}) + c_3(\tilde{d}_{k+1}) \cdot E'_k(\tilde{d}^{n-1}_{k+1}) && (82) \end{aligned}$$

$$\begin{aligned} B'_{k+1}(\tilde{d}_{k+1}) = {} & l' \cdot B'_{k+1}(\lambda) + s \cdot B'_{k+1}(\tilde{d}^2_{k+1}) \\ & + c_2(\tilde{d}_{k+2}) \cdot C'_{k+1}(\tilde{d}_{k+1} * \tilde{d}_{k+2}) \\ & + c'_3(\tilde{d}_{k+1}) \cdot J'_k(\lambda) && (83) \end{aligned}$$

$$\begin{aligned} B'_{k+1}(\tilde{d}^n_{k+1}) = {} & l \cdot B'_{k+1}(\tilde{d}^{n-1}_{k+1}) + s \cdot B'_{k+1}(\tilde{d}^{n+1}_{k+1}) && (n \geq 2) \\ & + c_2(\tilde{d}_{k+2}) \cdot C'_{k+1}(\tilde{d}^n_{k+1} * \tilde{d}_{k+2}) \\ & + c_3(\tilde{d}_{k+1}) \cdot J'_k(\tilde{d}^{n-1}_{k+1}) && (84) \end{aligned}$$

$$\begin{aligned} B'_{k+1}(\tilde{d}_{k+1} * \tilde{d}^m_{k+2}) = {} & l' \cdot B'_{k+1}(\tilde{d}^m_{k+2}) + l \cdot B'_{k+1}(\tilde{d}_{k+1} * \tilde{d}^{m-1}_{k+2}) \\ & + s \cdot B'_{k+1}(\tilde{d}^2_{k+1} * \tilde{d}^m_{k+2}) + s \cdot B'_{k+1}(\tilde{d}_{k+1} * \tilde{d}^{m+1}_{k+2}) \\ & + c_2(\tilde{d}_{k+2}) \cdot C'_{k+1}(\tilde{d}_{k+1} * \tilde{d}^{m+1}_{k+2}) \\ & + c'_3(\tilde{d}_{k+1}) \cdot J'_k(\tilde{d}^m_{k+2}) && (85) \end{aligned}$$

$$\begin{aligned} B'_{k+1}(\tilde{d}^n_{k+1} * \tilde{d}^m_{k+2}) = {} & l \cdot B'_{k+1}(\tilde{d}^{n-1}_{k+1} * \tilde{d}^m_{k+2}) + l \cdot B'_{k+1}(\tilde{d}^n_{k+1} * \tilde{d}^{m-1}_{k+2}) && (n \geq 2) \\ & + s \cdot B'_{k+1}(\tilde{d}^{n+1}_{k+1} * \tilde{d}^m_{k+2}) + s \cdot B'_{k+1}(\tilde{d}^n_{k+1} * \tilde{d}^{m+1}_{k+2}) \\ & + c_2(\tilde{d}_{k+2}) \cdot C'_{k+1}(\tilde{d}^n_{k+1} * \tilde{d}^{m+1}_{k+2}) \\ & + c_3(\tilde{d}_{k+1}) \cdot J'_k(\tilde{d}^{n-1}_{k+1} * \tilde{d}^m_{k+2}) && (86) \end{aligned}$$

$$C'_k(\lambda) = t \cdot B'_k(\lambda) \tag{87}$$

$$\begin{aligned} C'_k(\tilde{d}^n_{k+1}) = {} & l \cdot C'_k(\tilde{d}^{n-1}_{k+1}) + s \cdot C'_k(\tilde{d}^{n+1}_{k+1}) && (n \geq 1) \\ & + t \cdot B'_k(\tilde{d}^n_{k+1}) + c_3(\tilde{d}_{k+1}) \cdot D'_k(\tilde{d}^{n-1}_{k+1}) && (88) \end{aligned}$$

$$C'_{k+1}(\tilde{d}_{k+1}) = l' \cdot C'_{k+1}(\lambda) + s \cdot C'_{k+1}(\tilde{d}^2_{k+1})$$

$$
\begin{aligned}
&\quad + t \cdot B'_{k+1}(\tilde{d}_{k+1}) + c'_3(\tilde{d}_{k+1}) \cdot G'_k(\lambda) && (89)\\
C'_{k+1}(\tilde{d}^n_{k+1}) &= l \cdot C'_{k+1}(\tilde{d}^{n-1}_{k+1}) + s \cdot C'_{k+1}(\tilde{d}^{n+1}_{k+1}) && (n \geq 2)\\
&\quad + t \cdot B'_{k+1}(\tilde{d}^n_{k+1}) + c_3(\tilde{d}_{k+1}) \cdot G'_k(\tilde{d}^{n-1}_{k+1}) && (90)\\
C'_{k+1}(\tilde{d}_{k+1} * \tilde{d}^m_{k+2}) &= l' \cdot C'_{k+1}(\tilde{d}^m_{k+2}) + l \cdot C'_{k+1}(\tilde{d}_{k+1} * \tilde{d}^{m-1}_{k+2})\\
&\quad + s \cdot C'_{k+1}(\tilde{d}^2_{k+1} * \tilde{d}^m_{k+2}) + s \cdot C'_{k+1}(\tilde{d}_{k+1} * \tilde{d}^{m+1}_{k+2})\\
&\quad + t \cdot B'_{k+1}(\tilde{d}_{k+1} * \tilde{d}^m_{k+2})\\
&\quad + c'_3(\tilde{d}_{k+1}) \cdot G'_k(\tilde{d}^m_{k+2}) && (91)\\
C'_{k+1}(\tilde{d}^n_{k+1} * \tilde{d}^m_{k+2}) &= l \cdot C'_{k+1}(\tilde{d}^{n-1}_{k+1} * \tilde{d}^m_{k+2}) + l \cdot C'_{k+1}(\tilde{d}^n_{k+1} * \tilde{d}^{m-1}_{k+2}) && (n \geq 2)\\
&\quad + s \cdot C'_{k+1}(\tilde{d}^{n+1}_{k+1} * \tilde{d}^m_{k+2}) + s \cdot C'_{k+1}(\tilde{d}^n_{k+1} * \tilde{d}^{m+1}_{k+2})\\
&\quad + t \cdot B'_{k+1}(\tilde{d}^n_{k+1} * \tilde{d}^m_{k+2})\\
&\quad + c_3(\tilde{d}_{k+1}) \cdot G'_k(\tilde{d}^{n-1}_{k+1} * \tilde{d}^m_{k+2}) && (92)
\end{aligned}
$$

$$
\begin{aligned}
D'_k(\lambda) &= t \cdot E'_k(\lambda) + s_5(d_{k+1}) \cdot F'_k(\lambda) && (93)\\
D'_k(\tilde{d}^n_{k+1}) &= l \cdot D'_k(\tilde{d}^{n-1}_{k+1}) + s \cdot D'_k(\tilde{d}^{n+1}_{k+1}) && (n \geq 1)\\
&\quad + t \cdot E'_k(\tilde{d}^n_{k+1}) + s_5(d_{k+1}) \cdot F'_k(\tilde{d}^n_{k+1}) && (94)
\end{aligned}
$$

$$
\begin{aligned}
E'_k(\lambda) &= c_2(\tilde{d}_{k+1}) \cdot D'_k(\tilde{d}_{k+1}) + s_5(d_{k+1}) \cdot H'_k(\lambda) && (95)\\
E'_k(\tilde{d}^n_{k+1}) &= l \cdot E'_k(\tilde{d}^{n-1}_{k+1}) + s \cdot E'_k(\tilde{d}^{n+1}_{k+1}) && (n \geq 1)\\
&\quad + c_2(\tilde{d}_{k+1}) \cdot D'_k(\tilde{d}^{n+1}_{k+1}) + s_5(d_{k+1}) \cdot H'_k(\tilde{d}^n_{k+1}) && (96)
\end{aligned}
$$

$$
\begin{aligned}
F'_k(\lambda) &= t \cdot H'_k(\lambda) + c_4(\overline{k}) \cdot A'_{k+1}(\lambda) && (97)\\
F'_k(\tilde{d}^n_{k+1}) &= l \cdot F'_k(\tilde{d}^{n-1}_{k+1}) + s \cdot F'_k(\tilde{d}^{n+1}_{k+1}) && (n \geq 1)\\
&\quad + t \cdot H'_k(\tilde{d}^n_{k+1}) + c_4(\overline{k}) \cdot A'_{k+1}(\tilde{d}^n_{k+1}) && (98)
\end{aligned}
$$

$$
\begin{aligned}
G'_k(\lambda) &= t \cdot J'_k(\lambda) + c_4(\overline{k}) \cdot B'_{k+1}(\lambda) && (99)\\
G'_k(\tilde{d}^m_{k+2}) &= l \cdot G'_k(\tilde{d}^{m-1}_{k+2}) + s \cdot G'_k(\tilde{d}^{m+1}_{k+2})\\
&\quad + t \cdot J'_k(\tilde{d}^m_{k+2}) + c_4(\overline{k}) \cdot B'_{k+1}(\tilde{d}^m_{k+2}) && (100)\\
G'_k(\tilde{d}_{k+1}) &= l' \cdot G'_k(\lambda) + s \cdot G'_k(\tilde{d}^2_{k+1})\\
&\quad + t \cdot J'_k(\tilde{d}_{k+1}) + c_4(\overline{k}) \cdot B'_{k+1}(\tilde{d}_{k+1}) && (101)\\
G'_k(\tilde{d}^n_{k+1}) &= l \cdot G'_k(\tilde{d}^{n-1}_{k+1}) + s \cdot G'_k(\tilde{d}^{n+1}_{k+1}) && (n \geq 2)\\
&\quad + t \cdot J'_k(\tilde{d}^n_{k+1}) + c_4(\overline{k}) \cdot B'_{k+1}(\tilde{d}^n_{k+1}) && (102)\\
G'_k(\tilde{d}_{k+1} * \tilde{d}^m_{k+2}) &= l' \cdot G'_k(\tilde{d}^m_{k+2}) + l \cdot G'_k(\tilde{d}_{k+1} * \tilde{d}^{m-1}_{k+2})\\
&\quad + s \cdot G'_k(\tilde{d}^2_{k+1} * \tilde{d}^m_{k+2}) + s \cdot G'_k(\tilde{d}_{k+1} * \tilde{d}^{m+1}_{k+2})\\
&\quad + t \cdot J'_k(\tilde{d}_{k+1} * \tilde{d}^m_{k+2})\\
&\quad + c_4(\overline{k}) \cdot B'_{k+1}(\tilde{d}_{k+1} * \tilde{d}^m_{k+2}) && (103)\\
G'_k(\tilde{d}^n_{k+1} * \tilde{d}^m_{k+2}) &= l \cdot G'_k(\tilde{d}^{n-1}_{k+1} * \tilde{d}^m_{k+2}) + l \cdot G'_k(\tilde{d}^n_{k+1} * \tilde{d}^{m-1}_{k+2}) && (n \geq 2)
\end{aligned}
$$

$$
\begin{aligned}
&+ s \cdot G'_k(\tilde{d}^{n+1}_{k+1} * \tilde{d}^m_{k+2}) + s \cdot G'_k(\tilde{d}^n_{k+1} * \tilde{d}^{m+1}_{k+2})\\
&+ t \cdot J'_k(\tilde{d}^n_{k+1} * \tilde{d}^m_{k+2})\\
&+ c_4(\overline{k}) \cdot B'_{k+1}(\tilde{d}^n_{k+1} * \tilde{d}^m_{k+2}) && (104)
\end{aligned}
$$

$$
\begin{aligned}
H'_k(\lambda) &= c_2(\tilde{d}_{k+1}) \cdot F'_k(\tilde{d}_{k+1}) && (105)\\
H'_k(\tilde{d}^n_{k+1}) &= l \cdot H'_k(\tilde{d}^{n-1}_{k+1}) + s \cdot H'_k(\tilde{d}^{n+1}_{k+1}) && (n \geq 1)\\
&\quad + c_2(\tilde{d}_{k+1}) \cdot F'_k(\tilde{d}^{n+1}_{k+1}) && (106)
\end{aligned}
$$

$$
\begin{aligned}
I'_k(\lambda) &= c_1(d_{k+2}) \cdot J'_k(\lambda) && (107)\\
I'_k(\tilde{d}^n_{k+1}) &= l \cdot I'_k(\tilde{d}^{n-1}_{k+1}) + s \cdot I'_k(\tilde{d}^{n+1}_{k+1}) && (n \geq 1)\\
&\quad + c_1(d_{k+2}) \cdot J'_k(\tilde{d}^n_{k+1}) && (108)
\end{aligned}
$$

$$
\begin{aligned}
J'_k(\lambda) &= c_2(\tilde{d}_{k+2}) \cdot G'_k(c_2(\tilde{d}_{k+2})) && (109)\\
J'_k(\tilde{d}^m_{k+2}) &= l \cdot J'_k(\tilde{d}^{m-1}_{k+2}) + s \cdot J'_k(\tilde{d}^{m+1}_{k+2})\\
&\quad + c_2(\tilde{d}_{k+2}) \cdot G'_k(\tilde{d}^{m+1}_{k+2}) && (110)\\
J'_k(\tilde{d}_{k+1}) &= l' \cdot J'_k(\lambda) + s \cdot J'_k(\tilde{d}^2_{k+1})\\
&\quad + c_2(\tilde{d}_{k+2}) \cdot G'_k(\tilde{d}_{k+1} * \tilde{d}_{k+2}) && (111)\\
J'_k(\tilde{d}^n_{k+1}) &= l \cdot J'_k(\tilde{d}^{n-1}_{k+1}) + s \cdot J'_k(\tilde{d}^{n+1}_{k+1}) && (n \geq 2)\\
&\quad + c_2(\tilde{d}_{k+2}) \cdot G'_k(\tilde{d}^n_{k+1} * \tilde{d}_{k+2}) && (112)\\
J'_k(\tilde{d}_{k+1} * \tilde{d}^m_{k+2}) &= l' \cdot J'_k(\tilde{d}^m_{k+2}) + l \cdot J'_k(\tilde{d}_{k+1} * \tilde{d}^{m-1}_{k+2})\\
&\quad + s \cdot J'_k(\tilde{d}^2_{k+1} * \tilde{d}^m_{k+2}) + s \cdot J'_k(\tilde{d}_{k+1} * \tilde{d}^{m+1}_{k+2})\\
&\quad + c_2(\tilde{d}_{k+2}) \cdot G'_k(\tilde{d}_{k+1} * \tilde{d}^{m+1}_{k+2}) && (113)\\
J'_k(\tilde{d}^n_{k+1} * \tilde{d}^m_{k+2}) &= l \cdot J'_k(\tilde{d}^{n-1}_{k+1} * \tilde{d}^m_{k+2}) + l \cdot J'_k(\tilde{d}^n_{k+1} * \tilde{d}^{m-1}_{k+2}) && (n \geq 2)\\
&\quad + s \cdot J'_k(\tilde{d}^{n+1}_{k+1} * \tilde{d}^m_{k+2}) + s \cdot J'_k(\tilde{d}^n_{k+1} * \tilde{d}^{m+1}_{k+2})\\
&\quad + c_2(\tilde{d}_{k+2}) \cdot J'_k(\tilde{d}^n_{k+1} * \tilde{d}^{m+1}_{k+2}) && (114)
\end{aligned}
$$

Let E' be the guarded recursive specification consisting of the equations (79)–(114) and let E be the guarded recursive specification consisting of the same equations but without the extra acutes on the atomic actions. According to *RDP* there is a solution for the guarded recursive specification E'. We will denote this solution as follows.

$$a'_k(\lambda), a'_{k+1}(\tilde{d}^n_{k+1}), \ldots, j'_k(\tilde{d}^n_{k+1} * \tilde{d}^m_{k+2}).$$

We will show that we may use these processes in order to prove the theorem. Therefore, we will introduce a linear unary operator. Let $\{r\}$ be a set of function names and consider the following linear functional specification $E(\{r\})$.

$$
\begin{aligned}
E(\{r\}) = &\left\{\, r(l') = l, r\big(c'_3(x)\big) = c_3(x) : x \in E \,\right\}\\
&\cup \left\{\, r(a) = a : a \in A \setminus \{l', c'_3(x) : x \in E\} \,\right\}\\
&\cup \left\{\, r(a \cdot x) = r(a) \cdot r(x) : a \in A \,\right\}.
\end{aligned}
$$

In accordance with *ODP* there is a solution for this system, say ϱ. It will be clear that

$$\varrho\big(a'_k(\lambda)\big), \varrho\big(a'_{k+1}(\tilde{d}^n_{k+1})\big), \ldots, \varrho\big(j'_k(\tilde{d}^n_{k+1} * \tilde{d}^m_{k+2})\big)$$

is a solution for the guarded recursive specification E without accented atomic actions. But it is also evident that

$$\partial_H\big(A_k(\lambda)\big), \partial_H\big(A_{k+1}(\tilde{d}^n_{k+1})\big), \ldots, \partial_H\big(J_k(\tilde{d}^n_{k+1} * \tilde{d}^m_{k+2})\big)$$

is a solution for the guarded recursive specification E. So with the aid of *RSP* we conclude that these solutions are equal and we find, in particular,

$$\varrho\big(a'_0(\lambda)\big) = \partial_H\big(A_0(\lambda)\big). \tag{115}$$

Let $I' = I \cup \{l\} \cup \{c'_3(x) : x \in E\}$ be a subset of the set of atomic actions A. We will prove that $\tau_{I'} = \tau_I \circ \varrho$. Let $\{t\}$ be a functon name and let $E(\{t\})$ be the defining linear functional specification for $\tau_{I'}$.

$$\begin{aligned} E(\{t\}) = \big\{\, & t(a) = a : a \in A \setminus I' \,\big\} \\ & \cup \big\{\, t(a) = \tau : a \in I' \,\big\} \\ & \cup \big\{\, t(a \cdot x) = t(a) \cdot t(x) : a \in A \,\big\}. \end{aligned}$$

It will be clear that $\tau_{I'}$ solves this system. It is not hard to see that $\tau_I \circ \varrho$ is also a solution for the linear functional specification $E(\{t\})$. So in accord with *OSP* we find that these solutions must be equal and $\tau_{I'} = \tau_I \circ \varrho$. With this in mind we can deduce the following relation between a'_0 and A_0 using equation (115).

$$\begin{aligned} \tau_{I'}\big(a'_0(\lambda)\big) &= \tau_I \circ \varrho\big(a'_0(\lambda)\big) \\ &= \tau_I \circ \partial_H\big(A_0(\lambda)\big). \end{aligned}$$

So in order to prove theorem 4.4.16 it will be sufficient to deduce that the following equation is valid.

$$\tau_{I'}\big(a'_0(\lambda)\big) = \tau \cdot \prod_{i=1}^{\infty} s_5(d_i). \tag{116}$$

The last part of this proof will consist of the verification of equation (116). So for convenience sake we will stipulate the structure of this last part. We will make an assumption on a *DE*-trace for a fixed $p \geq 1$. Then we will state a conclusion on an *FH*-trace that can be derived from this assumption. Then the actual derivation of this will follow. It will be easy to see that the assumption on this *DE*-trace is valid for $p = 1$. With the aid of this we can set off a chain reaction and in the end we will find that both the assumption on the *DE*-trace and the conclusion on the *FH*-trace are true statements by themselves. With this knowledge we can make a small calculation on a single BC-trace. Then a straightforward deduction in which we will use GIP will end the proof of (116).

Let $J = \{s_5(d) : d \in D\}$ be a subset of the set of atomic actions A. Let $\beta_1 = \beta(I', J)$. Let $p \geq 1$ be arbitrary but fixed. Suppose that the following holds for all $k \geq 0$ and $n \geq 1$.

$$\pi_J^p \circ \beta_1\big(d_k'(\lambda)\big) = t \cdot \pi_J^p \circ \beta_1\big(e_k'(\lambda)\big) + \prod_{i=1}^{p} s_5(d_{k+i}) \tag{117}$$

$$\begin{aligned}\pi_J^p \circ \beta_1\big(d_k'(\tilde{d}_{k+1}^n)\big) = {} & l \cdot \pi_J^p \circ \beta_1\big(d_k'(\tilde{d}_{k+1}^{n-1})\big) + s \cdot \pi_J^p \circ \beta_1\big(d_k'(\tilde{d}_{k+1}^{n+1})\big) \\ & + t \cdot \pi_J^p \circ \beta_1\big(e_k'(\tilde{d}_{k+1}^n)\big) + \prod_{i=1}^{p} s_5(d_{k+i})\end{aligned} \tag{118}$$

$$\pi_J^p \circ \beta_1\big(e_k'(\lambda)\big) = c_2(\tilde{d}_{k+1}) \cdot \pi_J^p \circ \beta_1\big(e_k'(\tilde{d}_{k+1})\big) + \prod_{i=1}^{p} s_5(d_{k+i}) \tag{119}$$

$$\begin{aligned}\pi_J^p \circ \beta_1\big(e_k'(\tilde{d}_{k+1}^n)\big) = {} & l \cdot \pi_J^p \circ \beta_1\big(e_k'(\tilde{d}_{k+1}^{n-1})\big) + s \cdot \pi_J^p \circ \beta_1\big(e_k'(\tilde{d}_{k+1}^{n+1})\big) \\ & + c_2(\tilde{d}_{k+1}) \cdot \pi_J^p \circ \beta_1\big(d_k'(\tilde{d}_{k+1}^{n+1})\big) \\ & + \prod_{i=1}^{p} s_5(d_{k+i}).\end{aligned} \tag{120}$$

Then the following can be derived from these four equations for all $k, n \geq 0$:

$$\left.\begin{array}{l}\pi_J^p \circ \tau_{I'}\big(f_k'(\tilde{d}_{k+1}^n)\big) \\ \pi_J^p \circ \tau_{I'}\big(h_k'(\tilde{d}_{k+1}^n)\big)\end{array}\right\} = \tau \cdot \prod_{i=1}^{p} s_5(d_{k+1+i}). \tag{121}$$

In order to derive this we will first verify a similar result from the assumptions on double BC-traces and the JG-traces. In the subsequent display we will state what we are going to deduce from the assumptions for all $k, u, m \geq 0$.

$$\left.\begin{array}{l}\pi_J^p \circ \tau_{I'}\big(b_{k+1}'(\tilde{d}_{k+1}^u * \tilde{d}_{k+2}^m)\big) \\ \pi_J^p \circ \tau_{I'}\big(c_{k+1}'(\tilde{d}_{k+1}^u * \tilde{d}_{k+2}^{m+1})\big) \\ \pi_J^p \circ \tau_{I'}\big(g_k'(\tilde{d}_{k+1}^u * \tilde{d}_{k+2}^{m+1})\big) \\ \pi_J^p \circ \tau_{I'}\big(j_k'(\tilde{d}_{k+1}^u * \tilde{d}_{k+2}^m)\big)\end{array}\right\} = \tau \cdot \prod_{i=1}^{p} s_5(d_{k+1+i}). \tag{122}$$

In the former theorems we verified this with induction on u. In fact, we proved it with induction on the number of old datum elements available. This number could not be increased once we entered a double BC-trace or a JG-trace. So it was convenient to use this induction on the floor height of the traces. If we had assumed that the queue could only duplicate "new" datum elements, we could still use the floor height induction argument. For, we would not be able to reach a higher floor once we entered a double *BC*-trace or a *JG*-trace. And the calculations in the previous theorems could have been reiterated. Now we are in a different situation: as soon as we enter a trace that contains just one old datum element we can reach every higher situated horizontal trace of the dense block, q.v. figure 4.10. However, the proof for the case $u = 0$ will be the same as in the former theorems since if $u = 0$ there are no old datum elements in the queue. We will use this result to calculate the general case $u \geq 0$. So let $u = 0$ and fix an arbitrary $k \geq 0$. Consider the following guarded recursive specification E_1.

$$\begin{aligned} E_1 = \{\ X_0^1 &= t \cdot Y_0^1 + Z_1 \\ Y_0^1 &= c_2(\tilde{d}_{k+2}) \cdot X_1^1 + Z_1 \\ X_n^1 &= l \cdot X_{n-1}^1 + s \cdot X_{n+1}^1 + t \cdot Y_n^1 + Z_1 \\ Y_n^1 &= l \cdot Y_{n-1}^1 + s \cdot Y_{n+1}^1 + c_2(\tilde{d}_{k+2}) \cdot X_{n+1}^1 + Z_1 \\ Z_1 &= \prod_{i=1}^{p} s_5(d_{k+1+i}) \ \big|\ n \geq 1 \ \}. \end{aligned}$$

Let $C_1 = \{X_n^1, Y_n^1 : n \geq 0\}$. Then C_1 is a conservative cluster of E_1 in I'. Since the set of exits $U(C_1) = \{Z_1\}$ is finite, we may apply CFAR^∞ and we find for all $n \geq 0$:

$$\tau_{I'}(X_n^1) = \tau_{I'}(Y_n^1) = \tau \cdot \prod_{i=1}^{p} s_5(d_{k+1+i}).$$

Because of equations (117)–(120) we see that if we put for all $n \geq 0$

$$\begin{aligned} X_n^1 &= \pi_J^p \circ \beta_1\big(d'_{k+1}(\tilde{d}_{k+2}^n)\big) \\ Y_n^1 &= \pi_J^p \circ \beta_1\big(e'_{k+1}(\tilde{d}_{k+2}^n)\big), \end{aligned}$$

this will solve the guarded recursive specification E_1. Now consider the following calculation.

$$\begin{aligned} \tau \cdot \prod_{i=1}^{p} s_5(d_{k+1+i}) &= \tau_{I'}(X_n^1) \\ &= \tau_{I'} \circ \pi_J^p \circ \beta_1\big(d'_{k+1}(\tilde{d}_{k+2}^n)\big) \\ &= \pi_J^p \circ \tau_{I'} \circ \beta_1\big(d'_{k+1}(\tilde{d}_{k+2}^n)\big) && \text{use (4.3.12)} \\ &= \pi_J^p \circ \tau_{I'}\big(d'_{k+1}(\tilde{d}_{k+2}^n)\big) && \text{see lemma (4.4.6)} \end{aligned}$$

Such a calculation can be made for e'_{k+1}, too. We find thus for all $n \geq 0$

$$\left.\begin{aligned} &\pi_J^p \circ \tau_{I'}\big(d'_{k+1}(\tilde{d}_{k+2}^n)\big) \\ &\pi_J^p \circ \tau_{I'}\big(e'_{k+1}(\tilde{d}_{k+2}^n)\big) \end{aligned}\right\} = \tau \cdot \prod_{i=1}^{p} s_5(d_{k+1+i}). \tag{123}$$

Now consider the following guarded recursive specification E_2.

$$
\begin{aligned}
E_2 = \{\ & X_0^2 = c_2(\tilde{d}_{k+2}) \cdot Y_1^2 \\
& Y_0^2 = t \cdot X_0^2 \\
& X_n^2 = l \cdot X_{n-1}^2 + s \cdot X_{n+1}^2 + c_2(\tilde{d}_{k+2}) \cdot Y_{n+1}^2 + Z_2 \\
& Y_n^2 = l \cdot Y_{n-1}^2 + s \cdot Y_{n+1}^2 + t \cdot X_n^2 + Z_2 \\
& Z_2 = c_3(\tilde{d}_{k+2}) \cdot \prod_{i=1}^{p} s_5(d_{k+1+i}) \mid n \geq 1 \,\}.
\end{aligned}
$$

Let $C_2 = \{X_n^2, Y_n^2 : n \geq 0\}$. Then C_2 is a conservative cluster of E_2 in I'. Since the set of exits $U(C_2) = \{Z_2\}$ is finite, we may apply $CFAR^\infty$ and we find for all $n \geq 0$:

$$\tau_{I'}(X_n^2) = \tau_{I'}(Y_n^2) = \tau \cdot \prod_{i=1}^{p} s_5(d_{k+1+i}).$$

Let $\beta_2 = \beta\bigl(I', \{c_3'(\tilde{d}_{k+2})\}\bigr)$. We will show that if we put for all $n \geq 0$

$$
\begin{aligned}
X_n^2 &= \pi_J^p \circ \beta_2\bigl(b_{k+1}'(\tilde{d}_{k+2}^n)\bigr) \\
Y_n^2 &= \pi_J^p \circ \beta_2\bigl(c_{k+1}'(\tilde{d}_{k+2}^n)\bigr),
\end{aligned}
\tag{124}
$$

this will solve the guarded recursive specification E_2. For $n = 0$ it is trivial to see that, with the aid of equations (81) and (87), this is true for the first two equations of E_2. Now let $n \geq 1$. We will verify the third equation of E_2.

$$
\begin{aligned}
\pi_J^p \circ \beta_2\bigl(b_{k+1}'(\tilde{d}_{k+2}^n)\bigr) &\overset{(82)}{=} l \cdot \pi_J^p \circ \beta_2\bigl(b_{k+1}'(\tilde{d}_{k+2}^{n-1})\bigr) \\
&\quad + s \cdot \pi_J^p \circ \beta_2\bigl(b_{k+1}'(\tilde{d}_{k+2}^{n+1})\bigr) \\
&\quad + c_2(\tilde{d}_{k+2}) \cdot \pi_J^p \circ \beta_2\bigl(c_{k+1}'(\tilde{d}_{k+2}^{n+1})\bigr) \\
&\quad + c_3(\tilde{d}_{k+2}) \cdot \pi_J^p \circ \tau_{I'}\bigl(e_{k+1}'(\tilde{d}_{k+2}^{n-1})\bigr) \\
&\overset{(123)}{=} l \cdot \pi_J^p \circ \beta_2\bigl(b_{k+1}'(\tilde{d}_{k+2}^{n-1})\bigr) \\
&\quad + s \cdot \pi_J^p \circ \beta_2\bigl(b_{k+1}'(\tilde{d}_{k+2}^{n+1})\bigr) \\
&\quad + c_2(\tilde{d}_{k+2}) \cdot \pi_J^p \circ \beta_2\bigl(c_{k+1}'(\tilde{d}_{k+2}^{n+1})\bigr) \\
&\quad + c_3(\tilde{d}_{k+2}) \cdot \tau \cdot \prod_{i=1}^{p} s_5(d_{k+1+i}).
\end{aligned}
$$

So we see that this solves the equation in E_2 for X_n^2. A similar calculation will yield the same result for the fourth equation of E_2. So the claim that (124) is a solution for E_2 is valid. We find

$$\tau \cdot \prod_{i=1}^{p} s_5(d_{k+1+i}) = \tau_{I'}(X_n^2)$$

$$\begin{aligned}
&= \tau_{I'} \circ \pi^p_J \circ \beta_2\big(b'_{k+1}(\tilde{d}^n_{k+2})\big) \\
&= \pi^p_J \circ \tau_{I'} \circ \beta_2\big(b'_{k+1}(\tilde{d}^n_{k+2})\big) \qquad \text{use (4.3.12)} \\
&= \pi^p_J \circ \tau_{I'}\big(b'_{k+1}(\tilde{d}^n_{k+2})\big). \quad \text{see lemma (4.4.6)}
\end{aligned}$$

Such a calculation can be made for c'_{k+1}, too. Summarizing we have for all $n \geq 0$

$$\left.\begin{aligned}
\pi^p_J \circ \tau_{I'}\big(b'_{k+1}(\tilde{d}^n_{k+2})\big) \\
\pi^p_J \circ \tau_{I'}\big(c'_{k+1}(\tilde{d}^n_{k+2})\big)
\end{aligned}\right\} = \tau \cdot \prod_{i=1}^{p} s_5(d_{k+1+i}). \tag{125}$$

This means that the first two equations of (122) are proved for $u = 0$ and our fixed $k \geq 0$. Consider the next guarded recursive specification E_3.

$$\begin{aligned}
E_3 = \big\{\ X^3_0 &= t \cdot Y^3_0 + Z_3 \\
Y^3_0 &= c_2(\tilde{d}_{k+2}) \cdot X^3_1 \\
X^3_m &= l \cdot X^3_{m-1} + s \cdot X^3_{m+1} + t \cdot Y^3_m + Z_3 \\
Y^3_m &= l \cdot Y^3_{m-1} + s \cdot Y^3_{m+1} + c_2(\tilde{d}_{k+2}) \cdot X^3_{m+1} \\
Z_3 &= c_4(\overline{k}) \cdot \prod_{i=1}^{p} s_5(d_{k+1+i}) \ \big|\ n \geq 1 \big\}.
\end{aligned}$$

Let $C_3 = \{X^3_m, Y^3_m : m \geq 0\}$. Then C_3 is a conservative cluster of E_3 in I'. Since the set of exits $U(C_3) = \{Z_3\}$ is finite, we may apply *CFAR*$^\infty$ and we find for all $m \geq 0$:

$$\tau_{I'}(X^3_m) = \tau_{I'}(Y^3_m) = \tau \cdot \prod_{i=1}^{p} s_5(d_{k+1+i}).$$

Let $\beta_3 = \beta\big(I', \{c_4(\overline{k})\}\big)$. It will not be difficult to see that if we put for all $m \geq 0$

$$\begin{aligned}
X^3_m &= \pi^p_J \circ \beta_3\big(g'_k(\tilde{d}^m_{k+2})\big) \\
Y^3_m &= \pi^p_J \circ \beta_3\big(j'_k(\tilde{d}^m_{k+2})\big),
\end{aligned} \tag{126}$$

this will solve the system E_3. We will only verify the first equation for E_3. Consider thereto the following display.

$$\begin{aligned}
\pi^p_J \circ \beta_3\big(g'_k(\lambda)\big) &\overset{(99)}{=} t \cdot \pi^p_J \circ \beta_3\big(j'_k(\lambda)\big) + c_4(\overline{k}) \cdot \pi^p_J \circ \tau_{I'}\big(b'_{k+1}(\lambda)\big) \\
&\overset{(125)}{=} t \cdot \pi^p_J \circ \beta_3\big(j'_k(\lambda)\big) + c_4(\overline{k}) \cdot \tau \cdot \prod_{i=1}^{p} s_5(d_{k+1+i}).
\end{aligned}$$

The other equations can be handled in the same way while using equation (125). Thus, we find that (126) solves the guarded recursive specification E_3. When we use the fact that the generalized projection operator and the abstraction operator commute and the fact that the abstraction operator absorbs the selective abstraction operator β_3 we will find with a calculation that is similar to the ones we did for E_1 and E_2 that for all $m \geq 0$

$$\left.\begin{array}{l}\pi_J^p \circ \tau_{I'}\big(g_k'(\tilde{d}_{k+2}^m)\big)\\ \pi_J^p \circ \tau_{I'}\big(j_k'(\tilde{d}_{k+2}^m)\big)\end{array}\right\} = \tau \cdot \prod_{i=1}^{p} s_5(d_{k+1+i}).$$

Now we find that the second two equations of (122) are also correct so we see that the case $u = 0$ is proved for (122), since $k \geq 0$ was arbitrarily chosen. We will use this result to prove equation (122) for all $u \geq 0$. We can think of the dense block as a very large (conservative) cluster. The only way out for this cluster is by removing all the old data from the queue, that is, the case $u = 0$. Precisely the atomic actions that result in this situation are acuted. So if we make a β that transforms into the abstraction operator after passing the acuted actions we can use the result for $u = 0$. To achieve this consider the subsequent equations. The guarded recursive specification E_4 consists of these equations for all $n \geq 2$ and $m \geq 1$. Let $k \geq 0$ be fixed.

$$\begin{aligned}
X_{1,0}^4 &= l' \cdot Z_4 + s \cdot X_{2,0}^4 + c_2(\tilde{d}_{k+2}) \cdot Y_{1,1}^4 + c_3'(\tilde{d}_{k+1}) \cdot Z_4\\
X_{n,0}^4 &= l \cdot X_{n-1,0}^4 + s \cdot X_{n+1,0}^4 + c_2(\tilde{d}_{k+2}) \cdot Y_{n,1}^4 + c_3(\tilde{d}_{k+1}) \cdot T_{n-1,0}^4\\
X_{1,m}^4 &= l' \cdot Z_4 + l \cdot X_{1,m-1}^4 + s \cdot X_{2,m}^4 + s \cdot X_{1,m+1}^4\\
&\qquad + c_2(\tilde{d}_{k+2}) \cdot Y_{1,m+1}^4 + c_3'(\tilde{d}_{k+1}) \cdot Z_4\\
X_{n,m}^4 &= l \cdot X_{n-1,m}^4 + l \cdot X_{n,m-1}^4 + s \cdot X_{n+1,m}^4 + s \cdot X_{n,m+1}^4\\
&\qquad + c_2(\tilde{d}_{k+2}) \cdot Y_{n,m+1}^4 + c_3(\tilde{d}_{k+1}) \cdot T_{n-1,m}^4\\
Y_{1,0}^4 &= l' \cdot Z_4 + s \cdot Y_{2,0}^4 + t \cdot X_{1,0}^4 + c_3'(\tilde{d}_{k+1}) \cdot Z_4\\
Y_{n,0}^4 &= l \cdot Y_{n-1,0}^4 + s \cdot Y_{n+1,0}^4 + t \cdot X_{n,0}^4 + c_3(\tilde{d}_{k+1}) \cdot Z_{n-1,0}^4\\
Y_{1,m}^4 &= l' \cdot Z_4 + l \cdot Y_{1,m-1}^4 + s \cdot Y_{2,m}^4 + s \cdot Y_{1,m+1}^4\\
&\qquad + t \cdot X_{1,m}^4 + c_3'(\tilde{d}_{k+1}) \cdot Z_4\\
Y_{n,m}^4 &= l \cdot Y_{n-1,m}^4 + l \cdot Y_{n,m-1}^4 + s \cdot Y_{n+1,m}^4 + s \cdot Y_{n,m+1}^4\\
&\qquad + t \cdot X_{n,m}^4 + c_3(\tilde{d}_{k+1}) \cdot Z_{n-1,m}^4\\
Z_{1,0}^4 &= l' \cdot Z_4 + s \cdot Z_{2,0}^4 + t \cdot T_{1,0}^4 + c_4(\overline{k}) \cdot X_{1,0}^4\\
Z_{n,0}^4 &= l \cdot Z_{n-1,0}^4 + s \cdot Z_{n+1,0}^4 + t \cdot T_{n,0}^4 + c_4(\overline{k}) \cdot X_{n,0}^4\\
Z_{1,m}^4 &= l' \cdot Z_4 + l \cdot Z_{1,m-1}^4 + s \cdot Z_{2,m}^4 + s \cdot Z_{1,m+1}^4\\
&\qquad + t \cdot T_{1,m}^4 + c_4(\overline{k}) \cdot X_{1,m}^4\\
Z_{n,m}^4 &= l \cdot Z_{n-1,m}^4 + l \cdot Z_{n,m-1}^4 + s \cdot Z_{n+1,m}^4 + s \cdot Z_{n,m+1}^4\\
&\qquad + t \cdot T_{n,m}^4 + c_4(\overline{k}) \cdot X_{n,m}^4
\end{aligned}$$

$$\begin{aligned}
T^4_{1,0} &= l' \cdot Z_4 + s \cdot T^4_{2,0} + c_2(\tilde{d}_{k+2}) \cdot Z^4_{1,1}\\
T^4_{n,0} &= l \cdot T^4_{n-1,0} + s \cdot T^4_{n+1,0} + c_2(\tilde{d}_{k+2}) \cdot Z^4_{n,1}\\
T^4_{1,m} &= l' \cdot Z_4 + l \cdot T^4_{1,m-1} + s \cdot T^4_{2,m} + s \cdot T^4_{1,m+1}\\
&\qquad + c_2(\tilde{d}_{k+2}) \cdot Z^4_{1,m+1}\\
T^4_{n,m} &= l \cdot T^4_{n-1,m} + l \cdot T^4_{n,m-1} + s \cdot T^4_{n+1,m} + s \cdot T^4_{n,m+1}\\
&\qquad + c_2(\tilde{d}_{k+2}) \cdot Z^4_{n,m+1}\\
Z_4 &= \prod_{i=1}^{p} s_5(d_{k+1+i})
\end{aligned}$$

Let $C_4 = \{X^4_{n,m}, Y^4_{n,m}, Z^4_{n,m}, T^4_{n,m} : n \geq 1, m \geq 0\}$. The set of exits is

$$U(C_4) = \{l' \cdot Z_4, c'_3(\tilde{d}_{k+1}) \cdot Z_4\}.$$

It is not hard to see that C_4 is a conservative cluster of E_4 in I'; so since the set of exits is finite we may apply *CFAR*$^\infty$ and we find for all $n \geq 1$ and $m \geq 0$

$$\tau_{I'}(X^4_{n,m}) = \tau_{I'}(Y^4_{n,m}) = \tau_{I'}(Z^4_{n,m}) = \tau_{I'}(T^4_{n,m}) = \tau \cdot \prod_{i=1}^{p} s_5(d_{k+1+i}). \qquad (127)$$

Let $\beta_4 = \beta\big(I', \{l', c'_3(\tilde{d}_{k+1})\}\big)$. It will be not that hard to see that, if we put for all $n \geq 1$ and $m \geq 0$

$$\begin{aligned}
X^4_{n,m} &= \pi^p_J \circ \beta_4\big(b'_{k+1}(\tilde{d}^n_{k+1} * \tilde{d}^m_{k+2})\big)\\
Y^4_{n,m} &= \pi^p_J \circ \beta_4\big(c'_{k+1}(\tilde{d}^n_{k+1} * \tilde{d}^m_{k+2})\big)\\
Z^4_{n,m} &= \pi^p_J \circ \beta_4\big(g'_k(\tilde{d}^n_{k+1} * \tilde{d}^m_{k+2})\big)\\
T^4_{n,m} &= \pi^p_J \circ \beta_4\big(j'_k(\tilde{d}^n_{k+1} * \tilde{d}^m_{k+2})\big)
\end{aligned}$$

this will solve the guarded recursive specification E_4. To show the usage of the acuted atomic actions we will prove the first four equations of the guarded recursive specification E_4.

$$\begin{aligned}
X^4_{1,0} &= \pi^p_J \circ \beta_4\big(b'_{k+1}(\tilde{d}_{k+1})\big)\\
&= l' \cdot \pi^p_J \circ \tau_{I'}\big(b'_{k+1}(\lambda)\big) + s \cdot \pi^p_J \circ \beta_4\big(b'_{k+1}(\tilde{d}^2_{k+1})\big) && \text{use (83)}\\
&\qquad + c_2(\tilde{d}_{k+2}) \cdot \pi^p_J \circ \beta_4\big(c'_{k+1}(\tilde{d}_{k+1} * \tilde{d}_{k+2})\big)\\
&\qquad + c'_3(\tilde{d}_{k+1}) \cdot \pi^p_J \circ \tau_{I'}\big(j'_k(\lambda)\big)\\
&= l' \cdot Z_4 + s \cdot X^4_{2,0} + c_2(\tilde{d}_{k+2}) \cdot Y^4_{1,1}\\
&\qquad + c'_3(\tilde{d}_{k+1}) \cdot Z_4. && \text{use (122) for } u = 0
\end{aligned}$$

The second equation. Let $n \geq 2$. In equation (84) there are no acuted atomic actions so β_4 will not transform into an abstraction operator.

$$\begin{aligned} X^4_{n,0} &= \pi^p_J \circ \beta_4\big(b'_{k+1}(\tilde{d}^n_{k+1})\big) \\ &= l \cdot X^4_{n-1,0} + s \cdot X^4_{n+1,0} + c_2(\tilde{d}_{k+2}) \cdot Y^4_{n,1} + c_3(\tilde{d}_{k+1}) \cdot T^4_{n-1,0}. \end{aligned}$$

The third equation.

$$\begin{aligned} X^4_{1,m} &= \pi^p_J \circ \beta_4\big(b'_{k+1}(\tilde{d}_{k+1} * \tilde{d}^m_{k+2})\big) \\ &= l' \cdot \pi^p_J \circ \tau_{I'}\big(b'_{k+1}(\tilde{d}^m_{k+2})\big) && \text{use (85)} \\ &\qquad + l \cdot X^4_{1,m-1} + s \cdot X^4_{2,m} + s \cdot X^4_{1,m+1} \\ &\qquad + c_2(\tilde{d}_{k+2}) \cdot Y^4_{1,m+1} + c'_3(\tilde{d}_{k+1}) \cdot \pi^p_J \circ \tau_{I'}\big(j'_k(\tilde{d}^m_{k+2})\big) \\ &= l' \cdot Z_4 + l \cdot X^4_{1,m-1} + s \cdot X^4_{2,m} + s \cdot X^4_{1,m+1} && \text{use (122)} \\ &\qquad + c_2(\tilde{d}_{k+2}) \cdot Y^4_{1,m+1} + c'_3(\tilde{d}_{k+1}) \cdot Z_4 \end{aligned}$$

The fourth equation. In equation (86) there are no accented atomic actions so β_4 will not do anything. Just as with the second equation the verification of the fourth equation is trivial. The other equations are verified in exactly the same way as these first four. Recall that we are proving (122). We will prove that the first equation of (122) holds.

$$\begin{aligned} \tau \cdot \prod_{i=1}^{p} s_5(d_{k+1+i}) &= \tau_{I'}(X^4_{n,m}) && \text{with (127)} \\ &= \tau_{I'} \circ \pi^p_J \circ \beta_4\big(b'_{k+1}(\tilde{d}^n_{k+1} * \tilde{d}^m_{k+2})\big) \\ &= \pi^p_J \circ \tau_{I'} \circ \beta_4\big(b'_{k+1}(\tilde{d}^n_{k+1} * \tilde{d}^m_{k+2})\big) && \text{use (4.3.12)} \\ &= \pi^p_J \circ \tau_{I'}\big(b'_{k+1}(\tilde{d}^n_{k+1} * \tilde{d}^m_{k+2})\big) && \text{see (4.4.6)} \end{aligned}$$

Since this deduction is valid for all $n \geq 1$ and $m \geq 0$ we see that the first equation of (122) is correct. The other three are verified analogously. We have also seen that (122) is correct for $u = 0$ so this will end the proof of (122).

Now we will verify that from equations (117)–(120) we can derive (121). Choose a fixed $k \geq 0$. First, we will prove for all $n \geq 0$ for the I-nodes:

$$\pi^p_J \circ \tau_{I'}\big(i'_k(\tilde{d}^n_{k+1})\big) = \tau \cdot \prod_{i=1}^{p} s_5(d_{k+1+i}). \tag{128}$$

Consider thereto the following guarded recursive specification E_5.

$$\begin{aligned} E_5 = \Big\{\, X^5_0 &= c_1(d_{k+2}) \cdot \prod_{i=1}^{p} s_5(d_{k+1+i}) \\ X^5_n &= l \cdot X^5_{n-1} + s \cdot X^5_{n+1} + X^5_0 \;\Big|\; n \geq 1 \,\Big\}. \end{aligned}$$

Let $C_5 = \{X_n^5 : n \geq 1\}$ and $U(C_5) = \{X_0^5\}$. We see that C_5 is a conservative cluster for E_5 in I' hence, we may apply $CFAR^\infty$ and we find for all $n \geq 1$

$$\tau_{I'}(X_n^5) = \tau \cdot \prod_{i=1}^{p} s_5(d_{k+1+i}) = \tau_{I'}(X_0^5).$$

Let $\beta_5 = \beta\big(I', \{c_1(d_{k+2})\}\big)$. We will show that if we put for all $n \geq 0$

$$X_n^5 = \pi_J^p \circ \beta_5\big(i_k'(\tilde{d}_{k+1}^n)\big)$$

this will solve the system E_5. We begin with the first equation.

$$\begin{aligned} X_0^5 &= \pi_J^p \circ \beta_5\big(i_k'(\lambda)\big) \\ &= c_1(d_{k+2}) \cdot \pi_J^p \circ \tau_{I'}\big(j_k'(\lambda)\big) && \text{see (107)} \\ &= c_1(d_{k+2}) \cdot \tau \cdot \prod_{i=1}^{p} s_5(d_{k+1+i}). && \text{use (122)} \end{aligned}$$

Now let $n \geq 1$.

$$\begin{aligned} X_n^5 &= \pi_J^p \circ \beta_5\big(i_k'(\tilde{d}_{k+1}^n)\big) \\ &= l \cdot X_{n-1}^5 + s \cdot X_{n+1}^5 + c_1(d_{k+2}) \cdot \pi_J^p \circ \tau_{I'}\big(j_k'(\tilde{d}_{k+1}^n)\big) && \text{use (108)} \\ &= l \cdot X_{n-1}^5 + s \cdot X_{n+1}^5 + c_1(d_{k+2}) \cdot \tau \cdot \prod_{i=1}^{p} s_5(d_{k+1+i}) && \text{use (122)} \\ &= l \cdot X_{n-1}^5 + s \cdot X_{n+1}^5 + X_0^5. \end{aligned}$$

Using theorem (4.3.12) and lemma (4.4.6) we will find that equation (128) is correct.

With the use of (128) we are going to verify that a similar result holds for the A-nodes. For all $n \geq 1$ the following is valid

$$\pi_J^p \circ \tau_{I'}\big(a_{k+1}'(\tilde{d}_{k+1}^n)\big) = \tau \cdot \prod_{i=1}^{p} s_5(d_{k+1+i}). \tag{129}$$

Consider the subsequent guarded recursive specification E_6.

$$\begin{aligned} E_6 = \Big\{\ X_0^6 &= c_1(d_{k+2}) \cdot \prod_{i=1}^{p} s_5(d_{k+1+i}) \\ X_n^6 &= l \cdot X_{n-1}^6 + s \cdot X_{n+1}^6 + X_0^6 + Z_6 \\ Z_6 &= c_3(\tilde{d}_{k+1}) \cdot \prod_{i=1}^{p} s_5(d_{k+1+i}) \ \Big|\ n \geq 1 \Big\}. \end{aligned}$$

Let $C_6 = \{X_n^6 : n \geq 1\}$ and $U(C_6) = \{X_0^6, Z_6\}$. It is easy to see that C_6 is a conservative cluster for E_6 in I' so in accord with *CFAR*$^\infty$ we find for all $n \geq 1$

$$\tau_{I'}(X_n^6) = \tau \cdot \prod_{i=1}^{p} s_5(d_{k+1+i}) = \tau_{I'}(X_0^6).$$

Let $\beta_6 = \beta\big(I', \{c_1(d_{k+2}), c_3(\tilde{d}_{k+1})\}\big)$. It is not difficult to see that if we put for all $n \geq 0$

$$X_n^6 = \pi_J^p \circ \beta_6\big(a'_{k+1}(\tilde{d}_{k+1}^n)\big)$$

this will solve the specification E_6. We will show this for the second equation of E_6.

$$\begin{aligned}
X_n^6 &= \pi_J^p \circ \beta_6\big(a'_{k+1}(\tilde{d}_{k+1}^n)\big) \\
&= l \cdot X_{n-1}^6 + s \cdot X_{n+1}^6 + c_1(d_{k+2}) \cdot \pi_J^p \circ \tau_{I'}\big(b'_{k+1}(\tilde{d}_{k+1}^n)\big) \\
&\quad + c_3(\tilde{d}_{k+1}) \cdot \pi_J^p \circ \tau_{I'}\big(i'_k(\tilde{d}_{k+1}^{n-1})\big) && \text{see (80)} \\
&= l \cdot X_{n-1}^6 + s \cdot X_{n+1}^6 + X_0^6 && \text{because of (122)} \\
&\quad + c_3(\tilde{d}_{k+1}) \cdot \pi_J^p \circ \tau_{I'}\big(i'_k(\tilde{d}_{k+1}^{n-1})\big) \\
&= l \cdot X_{n-1}^6 + s \cdot X_{n+1}^6 + X_0^6 + Z_6. && \text{use (128)}
\end{aligned}$$

And using (4.3.12) and (4.4.6) we find that equation (129) is verified.

Now we are in a position to prove that equation (121) is correct. Consider thereto the following guarded recursive specification E_7.

$$\begin{aligned}
E_7 = \big\{\; X_0^7 &= t \cdot Y_0^7 + Z_7 \\
Y_0^7 &= c_2(\tilde{d}_{k+1}) \cdot X_1^7 \\
X_n^7 &= l \cdot X_{n-1}^7 + s \cdot X_{n+1}^7 + t \cdot Y_n^7 + Z_7 \\
Y_n^7 &= l \cdot Y_{n-1}^7 + s \cdot Y_{n+1}^7 + c_2(\tilde{d}_{k+1}) \cdot X_{n+1}^7 \\
Z_7 &= c_4(\overline{k}) \cdot \prod_{i=1}^{p} s_5(d_{k+1+i}) \;\big|\; n \geq 1 \,\big\}.
\end{aligned}$$

Let $C_7 = \{X_n^7, Y_n^7 : n \geq 0\}$ and $U(C_7) = \{Z_7\}$. We see that C_7 is a conservative cluster for E_7 in I' so we can apply *CFAR*$^\infty$ and we find for all $n \geq 0$

$$\tau_{I'}(X_n^7) = \tau_{I'}(Y_n^7) = \tau \cdot \prod_{i=1}^{p} s_5(d_{k+1+i}).$$

Let $\beta_7 = \beta\big(I', \{c_4(\overline{k})\}\big)$. If we put for all $n \geq 0$

$$\begin{aligned}
X_n^7 &= \pi_J^p \circ \beta_7\big(f'_k(\tilde{d}_{k+1}^n)\big) \\
Y_n^7 &= \pi_J^p \circ \beta_7\big(h'_k(\tilde{d}_{k+1}^n)\big)
\end{aligned}$$

this will solve the guarded recursive specification E_7. We will verify the first equation. The other equations are verified analogously.

$$\begin{aligned} X_0^7 &= \pi_J^p \circ \beta_7\big(f_k'(\lambda)\big) \\ &= t \cdot Y_0^7 + c_4(\overline{k}) \cdot \pi_J^p \circ \tau_{I'}\big(a_{k+1}'(\lambda)\big) && \text{because of (97)} \\ &= t \cdot Y_0^7 + Z_7. && \text{use (129)} \end{aligned}$$

Combining these facts while using theorem (4.3.12) and lemma (4.4.6) we find that equation (121) is correct under the assumption that equations (117)–(120) hold.

At this point we will prove that without these assumptions equation (121) is also valid. We will show this with induction on p. Let $p = 1$. We will show that the equations (117)–(120) are valid for $p = 1$. Let us first consider equation (117) for $p = 1$. With the aid of equation (93) we immediately see

$$\pi_J^1 \circ \beta_1\big(d_k'(\lambda)\big) = t \cdot \pi_J^1 \circ \beta_1\big(e_k'(\lambda)\big) + s_5(d_{k+1}).$$

This is precisely assumption (117). Now we will look at the second assumption (118). With the aid of equation (94) we see at once

$$\begin{aligned} \pi_J^1 \circ \beta_1\big(d_k'(\tilde{d}_{k+1}^n)\big) = l \cdot \pi_J^1 \circ \beta_1\big(d_k'(\tilde{d}_{k+1}^{n-1})\big) + s \cdot \pi_J^1 \circ \beta_1\big(d_k'(\tilde{d}_{k+1}^{n+1})\big) \\ + t \cdot \pi_J^1 \circ \beta_1\big(e_k'(\tilde{d}_{k+1}^n)\big) + s_5(d_{k+1}). \end{aligned}$$

This is just equation (118). The remaining two assumptions are verified in the same trivial way. This means that equation (121) is valid for $p = 1$, which concludes the basis of our induction. Now suppose that (121) is correct for one $p \geq 1$. Then we will prove it for $p + 1$. Thereto, we will show that the assumptions (117)–(120) are valid for $p + 1$. Let us take a look at the first assumption. With the aid of equation (93) we find easily

$$\begin{aligned} \pi_J^{p+1} \circ \beta_1\big(d_k'(\lambda)\big) &= t \cdot \pi_J^{p+1} \circ \beta_1\big(e_k'(\lambda)\big) + s_5(d_{k+1}) \cdot \pi_J^p \circ \tau_{I'}\big(f_k'(\lambda)\big) \\ &= t \cdot \pi_J^{p+1} \circ \beta_1\big(e_k'(\lambda)\big) \\ &\quad + s_5(d_{k+1}) \cdot \tau \cdot \prod_{i=1}^{p} s_5(d_{k+1+i}) && \text{induction} \\ &= t \cdot \pi_J^{p+1} \circ \beta_1\big(e_k'(\lambda)\big) + \prod_{i=1}^{p+1} s_5(d_{k+i}). \end{aligned}$$

This means that the first assumption is satisfied. Below we will show this for the second assumption.

$$\pi_J^{p+1} \circ \beta_1\big(d_k'(\tilde{d}_{k+1}^n)\big) = l \cdot \pi_J^{p+1} \circ \beta_1\big(d_k'(\tilde{d}_{k+1}^{n-1})\big) \qquad \text{use (94)}$$

$$
\begin{aligned}
&+ s \cdot \pi_J^{p+1} \circ \beta_1\big(d'_k(\tilde{d}_{k+1}^{n+1})\big)\\
&+ t \cdot \pi_J^{p+1} \circ \beta_1\big(e'_k(\tilde{d}_{k+1}^{n})\big)\\
&+ s_5(d_{k+1}) \cdot \pi_J^{p} \circ \tau_{I'}\big(f'_k(\tilde{d}_{k+1}^{n})\big)\\
= {}& l \cdot \pi_J^{p+1} \circ \beta_1\big(d'_k(\tilde{d}_{k+1}^{n-1})\big) + s \cdot \pi_J^{p+1} \circ \beta_1\big(d'_k(\tilde{d}_{k+1}^{n+1})\big)\\
&+ t \cdot \pi_J^{p+1} \circ \beta_1\big(e'_k(\tilde{d}_{k+1}^{n})\big)\\
&+ s_5(d_{k+1}) \cdot \tau \cdot \prod_{i=1}^{p} s_5(d_{k+1+i}) \qquad \text{induction}\\
= {}& l \cdot \pi_J^{p+1} \circ \beta_1\big(d'_k(\tilde{d}_{k+1}^{n-1})\big) + s \cdot \pi_J^{p+1} \circ \beta_1\big(d'_k(\tilde{d}_{k+1}^{n+1})\big)\\
&+ t \cdot \pi_J^{p+1} \circ \beta_1\big(e'_k(\tilde{d}_{k+1}^{n})\big) + \prod_{i=1}^{p+1} s_5(d_{k+i}).
\end{aligned}
$$

The other two assumptions can be treated in exactly the same way. So we find that all the assumptions are valid for $p+1$. But then it follows that equation (121) is proved for $p+1$. This concludes the induction step. Now that we have seen that (121) it is evident that the assumptions that we made are also correct by themselves. (For, the case $p = 1$ was already correct and for $p > 1$ we can reiterate, using (121), the above calculation.) With that result we are going to prove that the following holds for all $n \geq 0$.

$$
\left.\begin{aligned}
\pi_J^{p} \circ \tau_I\big(d'_k(\tilde{d}_{k+1}^{n})\big)\\
\pi_J^{p} \circ \tau_I\big(e'_k(\tilde{d}_{k+1}^{n})\big)
\end{aligned}\right\} = \tau \cdot \prod_{i=1}^{p} s_5(d_{k+i}). \tag{130}
$$

Consider the subsequent guarded recursive specification E_8.

$$
\begin{aligned}
E_8 = \big\{\, X_0^8 &= t \cdot Y_0^8 + Z_8\\
Y_0^8 &= c_2(\tilde{d}_{k+1}) \cdot X_1^8 + Z_8\\
X_n^8 &= l \cdot X_{n-1}^8 + s \cdot X_{n+1}^8 + t \cdot Y_n^8 + Z_8\\
Y_n^8 &= l \cdot Y_{n-1}^8 + s \cdot Y_{n+1}^8 + c_2(\tilde{d}_{k+1}) \cdot X_{n+1}^8 + Z_8\\
Z_8 &= \prod_{i=1}^{p} s_5(d_{k+i}) \;\big|\; n \geq 1 \,\big\}.
\end{aligned}
$$

We see at once from equations (117)–(120) and equation (121) that if we put for all $n \geq 0$

$$
\begin{aligned}
X_n^8 &= \pi_J^{p} \circ \beta_1\big(d'_k(\tilde{d}_{k+1}^{n})\big)\\
Y_n^8 &= \pi_J^{p} \circ \beta_1\big(e'_k(\tilde{d}_{k+1}^{n})\big),
\end{aligned}
$$

that this will solve the system E_8. Let $C_8 = \{X_n^8, Y_n^8 : n \geq 0\}$. The set of exits is $U(C_8) = \{Z_8\}$. We may apply CFAR^∞ and we find for all $n \geq 0$

$$
\tau_{I'}(X_n^8) = \tau_{I'}(Y_n^8) = \tau \cdot \prod_{i=1}^{p} s_5(d_{k+i}).
$$

A simple commutativity/absorption argument will yield that equation (130) is valid.

Now we will deduce the following result on a single *BC*-trace. For all $n \geq 0$

$$\left.\begin{array}{l}\pi_J^p \circ \tau_{I'}\big(b'_k(\tilde{d}_{k+1}^n)\big)\\ \pi_J^p \circ \tau_{I'}\big(c'_k(\tilde{d}_{k+1}^n)\big)\end{array}\right\} = \tau \cdot \prod_{i=1}^{p} s_5(d_{k+i}). \tag{131}$$

To achieve this, we will need one more guarded recursive specification E_9.

$$\begin{aligned} E_9 = \{\, X_0^9 &= c_2(\tilde{d}_{k+1}) \cdot Y_1^9 \\ Y_0^9 &= t \cdot X_0^9 \\ X_n^9 &= l \cdot X_{n-1}^9 + s \cdot X_{n+1}^9 + c_2(\tilde{d}_{k+1}) \cdot Y_{n+1}^9 + Z_9 \\ Y_n^9 &= l \cdot Y_{n-1}^9 + s \cdot Y_{n+1}^9 + t \cdot X_n^9 + Z_9 \\ Z_9 &= c_3(\tilde{d}_{k+1}) \cdot \prod_{i=1}^{p} s_5(d_{k+i}) \;\big|\; n \geq 1 \,\}. \end{aligned}$$

Let $C_9 = \{X_n^9, Y_n^9 : n \geq 0\}$. It will be clear that this is a conservative cluster for E_9 in I'. The set of exits is $U(C_9) = \{Z_9\}$. So if we apply *CFAR*$^\infty$ we find for all $n \geq 0$

$$\tau_{I'}(X_n^9) = \tau_{I'}(Y_n^9) = \tau \cdot \prod_{i=1}^{p} s_5(d_{k+i}).$$

Let $\beta_8 = \beta\big(I', \{c_3(\tilde{d}_{k+1})\}\big)$. With the aid of equations (81), (82), (87) and (88) together with the just derived result on the *DE*-traces (130) it is not hard to see that if we put for all $n \geq 0$

$$\begin{aligned} X_n^9 &= \pi_J^p \circ \beta_8\big(b'_k(\tilde{d}_{k+1}^n)\big) \\ Y_n^9 &= \pi_J^p \circ \beta_8\big(c'_k(\tilde{d}_{k+1}^n)\big), \end{aligned}$$

this is a solution for the guarded recursive specification E_9. So a straightforward calculation gives us that equation (131) is correct.

At this point we will make the final calculations to prove the theorem. Suppose that $k = 0$. We will show that equation (116) is correct. Let $p \geq 1$ be fixed and consider the deduction below.

$$\begin{aligned} \pi_J^p \circ \tau_{I'}\big(a'_0(\lambda)\big) &= \tau \cdot \pi_J^p \circ \tau_{I'}\big(b'_0(\lambda)\big) && \text{use (79)} \\ &= \tau \cdot \tau \cdot \prod_{i=1}^{p} s_5(d_i) && \text{because of (131)} \\ &= \tau \cdot \pi_J^p\Big(\prod_{i=1}^{\infty} s_5(d_i)\Big). \end{aligned}$$

In accord with *GIP* we may conclude that

$$\tau_{I'}\big(a_0'(\lambda)\big) = \tau \cdot \prod_{i=1}^{\infty} s_5(d_i),$$

which is exactly equation (116). We have also seen the relation between a_0' and A_0; see equation (115). And with the aid of this equation we conclude that

$$\tau_I \circ \partial_H\big(A_0(\lambda)\big) = \tau \cdot \prod_{i=1}^{\infty} s_5(d_i).$$

This ends the proof of 4.4.16.

4.5. Conclusions and further research

We have presented a theory in which it is possible to introduce linear unary operators and in which we can reason on these operators in a comfortable way. The theory that we developed is a generalization of the theory that is the subject of research in chapter 3. In there it was not yet possible to specify the projection operators. This is now incorporated in the theory. Moreover, the generalized induction principle *GIP* has been added to the theory that we studied in chapter 3, in which we already studied the two other principles *ODP* and *OSP*. The results of chapter 3 can also be verified in this more general framework. Therefore, we focused in this chapter on theorems concerning operators that could not be specified with the more restrictive form of the theory: the projection operators and linear unary operators with an exit possibility.

In section 4.4 we found some interesting applications to the subject of protocol verification. We showed in that section that *GIP* can be very useful in the area of protocol verification, especially for algebraic verification techniques, whereas *AIP* and its set of full projections is not very useful due to the fact that they do, in general, not commute with the abstraction operator. It will be clear that in this area more work can be done: we verified three protocols that were just hypothetical ones. We can make the three example protocols more symmetric by giving the two channel specifications the same behaviour; it will be not that hard to verify these protocols in the same way as we did for the asymmetric ones, although it can be quite a lenghty job. It will also not be a difficult task to verify the examples with a stronger correctness criterion: namely that these protocols are, in fact, one bit buffers.

We can still not specify the generalized state operator, see section 1.3, with a linear functional specification so further research on this subject is recommended. A way how to achieve this is mentioned briefly in section 3.7. When we are able to specify the generalized state operator and more protocol verifications of this type have been investigated, we can try to verify some sliding window protocols that have similar correctness criteria as the criterion that we used for our verifications; see [**24**].

Another interesting matter that can be worked out is the verification of theorem (4.4.12) and its more restrictive versions. In chapter 3 we can already see such a theorem: see theorem (3.6.1), page 135.

In this chapter we have not mentioned anything on the model theoretic side of $ACP_{\tau,u}$. In section 3.5 a model for the theory without abstraction ACP_u has been given and this can be generalized effortlessly to the more general situation that we studied in this chapter. However, on this subject still a lot of work has to be done.

If we look at the proofs that have been given in section 4.4 we can see that they consist of many steps that are almost the same. It might be the case that parts of these proofs can be done automagically and therefore it is also interesting to give formal specifications of these protocols with the use of an algebraic specification language. Examples of such languagese are *PSF*, see [**34**] and μCRL, see [**25**].

Samenvatting

De titel van dit proefschrift luidt in het Nederlands: lineaire unaire operatoren in de procesalgebra. Met behulp van de procesalgebra worden allerlei processen bestudeerd. Zo een proces kan zijn een koffieautomaat, een robot, een computer, een verkeerslicht of het breien van een sjaal. Om deze processen te beschrijven gaan we er vanuit dat ze een aantal elementaire acties kunnen uitvoeren. Te denken valt aan een muntje inwerpen, een verfspuit aanzetten, een karakter inlezen, het licht op groen zetten of één steek averecht breien. Deze acties worden wel atomair genoemd. Om nu het gedrag van een proces te beschrijven, gebruiken we behalve deze atomen ook een aantal bewerkingen. Zo wordt bijvoorbeeld het na elkaar uitvoeren van twee atomaire acties met een vermenigvuldigingsteken $\cdot$ aangegeven. Als we definiëren dat r staat voor één steek recht en a staat voor één steek averecht dan wordt één recht, één averecht als volgt genoteerd: $r \cdot a$. Zo zijn er nog een aantal van dit soort operaties. Als er gekozen moet worden tussen r en a wordt een $+$ gebruikt: $r + a$. Meestal wordt een proces gedefinieerd in termen van zichzelf. Dit heet recursie. Als we bijvoorbeeld een sjaal willen breien dan staat $B = r \cdot a \cdot B$ voor het proces dat één recht, één averecht doet en dan weer opnieuw begint.

Lineaire unaire operatoren worden in de procesalgebra gebruikt om specifieke problemen op te lossen. De werkwijze is meestal als volgt: er is een probleem dat opgelost moet worden maar de formalismen die op dat moment voor handen zijn, zijn niet toereikend. Derhalve wordt er naar een nieuw formalisme gezocht, hetgeen met zich meebrengt dat er allerlei standaardfeiten over dit formalisme bewezen moeten worden. In een aantal gevallen ligt de oplossing in het ad hoc definiëren van een of andere operator. Het formalisme waarmee het probleem aangepakt kan worden bestaat dan uit een basisformalisme aangevuld met de operator die het probleem op de juiste wijze modelleert teneinde tot een oplossing te geraken en natuurlijk een aantal stellingen met betrekking tot de nieuwe operator. Dit proefschrift vormt daar geen uitzondering op: in hoofdstuk 2 wordt een probleem opgelost door aan een bestaand formalisme een operator toe te voegen. Ook worden er aanvullende stellingen bewezen die nodig zijn voor de uiteindelijke oplossing ervan.

Om een voorbeeld van deze werkwijze te geven gaan we een stekentellertje inbouwen in ons breiproces B dat het aantal steken telt en elke 100 steken opnieuw begint te tellen. De register operator die in hoofdstuk 2 is behandeld leent zich daar uitstekend voor. We gaan eerst de acties r en a van een zogenaamd steekgetal voorzien: $r(k)$ en $a(k)$. Zo betekent $r(k)$ de $k$de steek recht.

Om het tellertje te kunnen ophogen definiëren we een functie f als volgt:

$$f(k) = k + 1 \pmod{101}.$$

Het getal k wordt veranderd in de rest van $k+1$ bij deling door 101. Dus

$$f(0) = 1, f(1) = 2, \ldots, f(99) = 100, f(100) = 0.$$

Ook zullen we een aantal pre-acties invoeren. Dit zijn weliswaar atomaire acties maar ze moeten gezien worden als rekengrootheden. Deze rekengrootheden zijn:

$$r\big(f(\uparrow)\big), a\big(f(\uparrow)\downarrow\big).$$

De pijltjes omhoog betekenen dat er een gegeven uit het geheugen van de register operator kan worden opgediept en een pijltje omlaag betekent dat er een gegeven in het geheugen geplaatst kan worden. Laten we nu naar het volgende proces kijken

$$X = r\big(f(\uparrow)\big) \cdot a\big(f(\uparrow)\downarrow\big) \cdot X.$$

Als we daar de register operator op los laten met als geheugeninhoud een nul samen met een operator τ_0 die de steken $r(0)$ en $a(0)$ met steekgetal nul onzichtbaar maakt, dan zien we dat $B = \tau_0 \circ [0](X)$ precies het proces is dat 100 rechte en 100 averechte steken telt en dan weer opnieuw begint te tellen. De operator τ_0 verandert de acties $r(0)$ en $a(0)$ in een speciale "onzichtbare" actie genaamd τ die de eigenschap heeft om te verdwijnen als hij wordt vooraf gegaan door een andere actie: $a(100) \cdot \tau = a(100)$. We berekenen $\tau_0 \circ [0](X)$:

$$\begin{aligned}
B &= \tau_0 \circ [0]\Big(r\big(f(\uparrow)\big) \cdot a\big(f(\uparrow)\downarrow\big) \cdot X\Big) \\
&= r(1) \cdot \tau_0 \circ [0]\Big(a\big(f(\uparrow)\downarrow\big) \cdot X\Big) \\
&= r(1) \cdot a(1) \cdot \tau_0 \circ [1](X) \\
&\ \vdots \\
&= r(1) \cdot a(1) \cdot r(2) \cdot a(2) \cdot \ldots \cdot r(100) \cdot a(100) \cdot \tau_0 \circ [100](X) \\
&= r(1) \cdot a(1) \cdot r(2) \cdot a(2) \cdot \ldots \cdot r(100) \cdot a(100) \cdot \tau_0\big(r(0) \cdot a(0) \cdot [0](X)\big) \\
&= r(1) \cdot a(1) \cdot r(2) \cdot a(2) \cdot \ldots \cdot r(100) \cdot a(100) \cdot \tau \cdot \tau \cdot \tau_0 \circ [0](X) \\
&= r(1) \cdot a(1) \cdot r(2) \cdot a(2) \cdot \ldots \cdot r(100) \cdot a(100) \cdot B.
\end{aligned}$$

En we zien dat er nu een stekentellerje is ingebouwd.

De rest van dit proefschrift bestaat uit pogingen om de hier boven geschetste werkwijze in een algemeen kader te plaatsen. In hoofdstuk 3 wordt een formalisme voorgesteld waarmee men de beschikking krijgt over een klasse van operatoren; de zogenaamde lineaire unaire operatoren. Een basisformalisme waar men vanuit zou kunnen gaan bij de "klassieke" werkwijze wordt verrijkt met een functieruimte waarin een groot aantal van de ad hoc operatoren leeft, die in het verleden reeds bedacht zijn. Er worden twee cruciale bewijsregels gelanceerd die het mogelijk maken om op een eenvoudige manier te kunnen redeneren over een deelklasse van de lineaire unaire operatoren. In dit hoofdstuk wordt aandacht besteed aan algemene stellingen over deze operatoren. Er wordt aan de hand van, uit de literatuur bekende, voorbeelden (die in verschillende theorieën zijn behandeld) plausibel gemaakt dat de voorgestelde theorie toereikend is om deze voorbeelden aan te pakken. Hiermee wordt eveneens aangetoond dat het voorgestelde formalisme unificerend werkt. Er wordt echter ook gewezen op een aantal beperkingen die er liggen bij het formalisme als zodanig.

We zullen nog éénmaal terugkomen op het breivoorbeeld. We hebben nu wel een tellertje gemaakt, maar we zouden na 100 steken een nieuwe pen willen kunnen opzetten en dan weer 100 steken breien. Met een n bedoelen we een nieuwe pen. We willen bereiken dat $B = (r \cdot a)^{100} \cdot n \cdot B$ hetgeen zoveel zeggen wil dat er na 100 keer één recht, één averecht met een nieuwe pen begonnen wordt en dan weer van voren af aan. We gaan een operator invoeren die het gedrag van $[0](X)$ zodanig aanpast dat we de nieuwe B krijgen. De operator die we willen invoeren moet voldoen aan een aantal voorwaarden. We zullen deze operator niet formeel invoeren zoals in hoofdstuk 3 is aangegeven, maar we zullen ons beperken tot de eigenschappen die de operator heeft. We noemen deze operator ρ van *renaming*. Als $k \neq 0$ dan gooien we gewoon het steekgetal weg: $\rho\big(r(k)\big) = r$ en $\rho\big(a(k)\big) = a$. Als $k = 0$ dan geven we $r(0)$ de naam n: $\rho\big(r(0)\big) = n$ van nieuwe pen en omdat we $a(0)$ niet nodig hebben ontdoen we hem weer van zijn identiteit door hem te veranderen in de onzichtbare actie τ, waar voor geldt: $n \cdot \tau = n$. We berekenen nu $\rho \circ [0](X)$:

$$\begin{aligned}\rho \circ [0](X) &= \rho\big(r(1) \cdot a(1) \cdot \ldots \cdot r(100) \cdot a(100) \cdot r(0) \cdot a(0) \cdot [0](X)\big)\\ &= (r \cdot a)^{100} \cdot \rho\big(r(0) \cdot a(0) \cdot [0](X)\big)\\ &= (r \cdot a)^{100} \cdot n \cdot \tau \cdot \rho \circ [0](X)\\ &= (r \cdot a)^{100} \cdot n \cdot \rho \circ [0](X).\end{aligned}$$

We zien dat het gedrag van $\rho \circ [0](X)$ uitdrukt dat er na 100 maal één recht, één averecht een nieuwe pen begonnen kan worden en dan weer van voren af aan.

In hoofdstuk 4 wordt één van de mogelijke bezwaren die men zou kunnen hebben tegen het formalisme in hoofdstuk 3 weggenomen. Het formalisme wordt op bepaalde plaatsen aangepast zodat er over een grotere deelklasse van operatoren kan worden geredeneerd met behulp van de bewijsregels die in hoofdstuk 3 uitgebreid aan de orde zijn geweest. Bovendien wordt er nog een andere bewijsregel aan de theorie toegevoegd waarmee het mogelijk is om inductieve bewijzen te beschouwen. In tegenstelling tot de gevolgde weg in hoofdstuk 3, ligt in dit hoofdstuk de nadruk op het toepassen van de tot nu toe ontwikkelde theorie. Er wordt een oud probleem in de procesalgebra opgelost en er wordt een methode geschetst om communicatieprotocollen door te rekenen, aan de hand van een drietal voorbeelden.

References

At this point we will catalogue all the references we made throughout this thesis. We will refer to the first possible source in which an item can be found. This is not always the best source. General references to the subject of process algebra and its applications are [**8**] for the process algebra and [**3**] for several applications.

[**1**] G. J. Akkerman, J. C. M. Baeten, *Term rewriting analysis in process algebra*, Report P9006, Programming Research Group, University of Amsterdam, 1990. To appear in CWI Quarterly, 1992.

[**2**] P. America, J. W. de Bakker, *Designing equivalent semantic models for process creation*, Theor. Comp. Sci. **60**, pp. 109–176, 1988.

[**3**] J. C. M. Baeten (Editor), *Applications of process algebra*, Cambridge Tracts in Theoretical Computer Science **17**, Cambridge University Press 1990.

[**4**] J. C. M. Baeten, J. A. Bergstra, *Global renaming operators in concrete process algebra*, Information and Computation **78**, pp. 205–245, 1988.

[**5**] J. C. M. Baeten, J. A. Bergstra, J. W. Klop, *Conditional axioms and α/β-calculus in process algebra*, in: Proc. IFIP Conf. on Formal Description of Programming Concepts III, Ebberup 1986 (M. Wirsing, ed.), North-Holland, Amsterdam, pp. 77–103, 1987.

[**6**] J. C. M. Baeten, J. A. Bergstra, J. W. Klop, *Syntax and defining equations for an interrupt mechanism in process algebra*, Fundamenta Informaticae **IX**, pp. 127–168, 1986.

[**7**] J. C. M. Baeten, R. J. van Glabbeek, *Another look at abstraction in process algebra*, In: Proc. 14th ICALP, Karlsruhe 1987 (Th. Ottmann, ed.), LNSC **267**, Springer Verlag, pp. 84–94, 1987.

[**8**] J. C. M. Baeten, W. P. Weijland, *Process algebra*, Cambridge Tracts in Theoretical Computer Science **18**, Cambridge University Press, 1990.

[**9**] J. W. Bakker, J. I. Zucker, *Denotational semantics of concurrency*, Proc. 14th ACM Symp. Theory of Comp., pp. 153–158, 1982.

[**10**] J. A. Bergstra, *A process creation mechanism in process algebra*, In [**3**], pp. 81–88.

[**11**] J. A. Bergstra, personal communication, March 1992.

[**12**] J. A. Bergstra, J. W. Klop, *Algebra of communicating processes with abstraction*, TCS **37**, 77–121, 1985.

[**13**] J. A. Bergstra, J. W. Klop, *Fair FIFO queues satisfy an algebraic criterion for protocol correctness*, Report CS-R8405, CWI Amsterdam, 1984.

[**14**] J. A. Bergstra, J. W. Klop, *Fixed point semantics in process algebras*, MC report IW **206**, Mathematical Centre, Amsterdam, 1982. Revised version: J. A. Bergstra, J. W. Klop, *A convergence theorem in process algebra*, CWI report CS-R8733, CWI Amsterdam, 1987.

[**15**] J. A. Bergstra, J. W. Klop, *Process Algebra: specification and verification in bisimulation semantics*, Math. & Comp. Sci. II (M. Hazewinkel, J. K. Lenstra, L. G. L. T. Meertens, eds.), CWI Monograph **4**, North-Holland, Amsterdam, pp. 61–94, 1986.

[**16**] J. A. Bergstra, J. W. Klop, *The algebra of recursively defined processes and the algebra of regular processes*, Proceedings 11th ICALP, Antwerpen 1984 (ed. J. Paredaens), Springer LNCS **172**, pp. 82–95, 1984.

[**17**] J. A. Bergstra, J. W. Klop, *Verification of an alternating bit protocol by means of process algebra*, In: Math. Methods of Spec. and Synthesis of Software Systems 1985 (eds. W. Bibel, K. P. Jantke), Math. Research **31**, Akademie-Verlag Berlin, pp. 9–23, 1985.

[**18**] J. J. Brunekreef, *A formal specification of three sliding window protocols*, Report P9102, Programming Research Group, University of Amsterdam, 1991. Revised version: J. J. Brunekreef, *A formal specification of three sliding window protocols (revised version)*, Report P9102b, Programming Research Group, University of Amsterdam, 1991.

[**19**] C. C. Chang, H. J. Keisler, *Model Theory*, Second Edition, North-Holland, Amsterdam, 1973.

[**20**] N. Dershowitz, *Termination of rewriting*, Journal of Symbolic Computation, **3**, pp. 69–116, 1987.

[**21**] R. J. van Glabbeek, *Bounded nondeterminism and the approximation induction principle in process algebra*, In: Proceedings STACS **87** (F. J. Brandenburg, G. Vidal-Naquet, M. Wirsing, eds.), LNCS **247**, Springer Verlag, pp. 336–347, 1987.

[**22**] R. J. van Glabbeek, F. W. Vaandrager, *Modular specifications in process algebra—with curious queues*, in: Algebraic Methods: Theory, Tools, and Applications (M. Wirsing, J. A. Bergstra, eds.), LNCS **394**, Springer Verlag, pp. 465–506, 1989.

[**23**] R. J. van Glabbeek, W. P. Weijland, *Branching time and abstraction in bisimulating semantics (extended abstract)*, Information Processing **89**, (G. X. Ritter, ed.), North-Holland, 613–618, Amsterdam 1989.

[**24**] R. A. Groenveld, *Verification of a sliding window protocol by means of process algebra*, Report P8701, Programming Research Group, University of Amsterdam, 1987.

[**25**] J. F. Groote, A. Ponse, *The syntax and semantics of μCRL*, CWI report CS-R9076, CWI Amsterdam, 1990. Also contained in: J. F. Groote, *Process algebra and structured operational semantics*, Ph D thesis, University of Amsterdam, 1991.

[**26**] M. Hennessy, *Algebraic theory of processes*, MIT Press, Cambridge Ma., 1988.

[**27**] C. A. R. Hoare, *Communicating Sequential Processes*, Prentice-Hall International, Englewood Cliffs, New Jersey, 1985.

[**28**] S. Kamin, J.-J. Lévy, *Two generalizations of the recursive path ordering*, Unpublished note, Department of Computer Science, University of Illinois, Urbana, IL, 1980.

[**29**] G. Karjoth, *Stepwise specification of a sliding-window protocol by means of process algebra*, Proc. 1988 Intl. Zürich Seminar on Digital Communications (IEEE Catalog Nr. 88TH0202-2), pp. 109–114, 1988.

[**30**] J. W. Klop, *Term Rewriting Systems*, CWI report CS-R9073, CWI Amsterdam, 1990. To appear in: Handbook of Logic in Computer Science (eds. S. Abramsky, D. Gabbay and T. Maibaum), Oxford University Press.

[**31**] J. C. Koomen, *A structure theory for communication network control*, Ph D thesis, Technical University Delft, 1982.

[**32**] E. Kranakis, *Fixed point equations with parameters in the projective limit model*, Inf. & Comp. **75**, pp. 264–288, 1987.

[**33**] H. T. Kung, C. E. Leiserson, *Systolic Arrays for VLSI,* Proc. of the symposium on sparse matrices computation, L. S. Duff et al. eds., Knoxville, Tenn. pp. 256–282, 1987.

[**34**] S. Mauw, G. J. Veltink, *A process specification formalism*, Fundamenta Informaticae **XIII**, pp. 85–139, 1990.

[**35**] K. Meinke, J. V. Tucker, *Universal Algebra*, Oxford University Press, To appear.

[**36**] R. Milner, *A calculus of communicating systems*, LNCS **92**, Springer Verlag, 1980.

[**37**] R. Milner, *Communication and Concurrency*, Prentice-Hall International, Englewood Cliffs, New Jersey, 1989.

[**38**] F. W. Vaandrager, *Algebraic techniques for concurrency and their application*, Ph D thesis, University of Amsterdam, 1990.

[**39**] F. W. Vaandrager, *Process algebra semantics of POOL*, In [**3**], pp. 173–236.

[**40**] F. W. Vaandrager, *Two simple protocols*, In [**3**], pp. 23–44.

[**41**] C. Verhoef, *An operator definition principle (for process algebras)*, Report P9105, Programming Research Group, University of Amsterdam, 1991.

[**42**] C. Verhoef, *On induction principles*, Report P9204, Programming Research Group, University of Amsterdam, 1992.

[**43**] C. Verhoef, *On the register operator*, Report P9003, Programming Research Group, University of Amsterdam, 1990.

[**44**] W. P. Weijland, *Synchrony and Asynchrony in Process Algebra*, Ph D Thesis, University of Amsterdam, 1989.