7

Parallel and communicating processes

7.1 Interleaving

So far, the focus has been on *sequential* processes: actions can be executed, or alternatives can be explored, starting from a single point of control. In this chapter, the step is taken towards the treatment of parallel or distributed systems: it is allowed that activities exist in parallel. Just allowing separate activity of different components is not enough. A genuine treatment of parallel activity requires in addition a description of interaction between parallel activities.

Suppose there are two sequential processes x and y that can execute actions, and choose alternatives, independently. The *merge* operator $\|$ denotes parallel composition. Thus, the parallel composition of x and y is denoted $x \parallel y$. To illustrate the intuition behind the algebraic treatment of parallel composition, consider an external observer $\mathcal{O}$ that observes process $x \parallel y$. Observations can be made of executions of actions. Assume that these observations are instantaneous. Then, it can be seen that the observations of actions of x and actions of y will be *merged* or *interleaved* in time.

Consider the example $a.0 \parallel b.0$. This process involves the execution of two actions, one from each component. Observer $\mathcal{O}$ might see the execution of a first, and then the execution of b. After this, no further activity is possible. On the other hand, observer $\mathcal{O}$ might see the execution of b first, then the execution of a followed by inaction. Finally, the observer might observe the two actions simultaneously.

The simultaneous observation of two actions is reserved for the interaction between the two processes, i.e., processes achieve interaction through *synchronization*; the occurrence of interaction between two processes is the result of the simultaneous execution of matching actions. By a judicious choice of

https://doi.org/10.1017/CBO9781139195003.011 Published online by Cambridge University Press

atomic actions, *communication* can be achieved with synchronization, i.e., a message can be passed from one process to another.

Returning to the simple example, assume the interaction that is achieved by the simultaneous execution of a and b is denoted by c. Summarizing the above discussions, the execution of $a.0 \parallel b.0$ has three possibilities: first a and then b, first b and then a, or c (denoting a and b together). In terms of an equation, this can be stated as follows. The merge operator $\parallel$ binds stronger than choice and weaker than action prefix.

$$a.0 \parallel b.0 = a.b.0 + b.a.0 + c.0.$$

The sketched approach to parallel composition is called *arbitrary interleaving* or *shuffle* in the literature. Note that the *observations* of the action executions are interleaved, not necessarily the actions themselves. So, the actions themselves can have duration (and will have duration in practice). It is also not necessarily the case that the first action has finished when the second action starts. If necessary, it can be made explicit that actions can overlap in time, by allowing observations of the beginning and the ending of an action:

$$begin_a.end_a.0 \parallel begin_b.end_b.0.$$

This process has $begin_a.begin_b.end_a.end_b.0$ as one of its possible execution sequences.

This chapter expands the theories BSP and TSP of the earlier chapters with the interleaving merge operator, presenting equational theories and the underlying models for communicating parallel processes.

7.2 An operational view

Before turning to an equational theory, parallel processing and communication is first studied from an operational point of view, to provide insight in some of the aspects involved.

Considering parallel processing without interaction or communication, the basic idea is that a component in a parallel composition can execute an action by itself, independent of other components. This can be described in a straightforward way by the following deduction rules (as defined in the context of term deduction systems), where x, x', y, and y' are arbitrary process terms and a is an action:

$$\frac{x \xrightarrow{a} x'}{x \parallel y \xrightarrow{a} x' \parallel y} \quad \text{and} \quad \frac{y \xrightarrow{a} y'}{x \parallel y \xrightarrow{a} x \parallel y'} .$$

Interaction is more involved. As mentioned, interaction is the simultaneous execution of actions that match in some sense. An example of communication

https://doi.org/10.1017/CBO9781139195003.011 Published online by Cambridge University Press

through synchronization has already been given in the previous section where the synchronized execution of actions a and b resulted in an action c. More concretely, one could consider a process that can perform a *send* action, and that is executing in parallel with another process that can perform a *receive* action. If it is assumed that these actions match, then the simultaneous executions of these two actions could be defined to result in a *communicate* action, thus achieving communication between parallel processes.

Communication resulting from the simultaneous execution of matching actions is *synchronous*, as opposed to asynchronous communication in which a message might be sent at one point in time and received at a different, later point in time. Synchronous communication is the basic form of communication in this book. Asynchronous communication can be achieved by explicitly specifying buffers or communication channels.

In elementary synchronous communication, every communication has two parts. Actions that may communicate can be specified through a *communication function* γ, that takes a pair of communicating actions and returns the result of the communication, which is also an action. Suppose an atomic action *communicate*(5) represents the communication of a data value 5. Intuitively, when communication is reliable, this communication action should be the result of the synchronized execution of a *send*(5) action and a *receive*(5) action; that is, the data value that is received should match the value that is sent. This can be defined as follows:

$$\gamma(send(5), receive(5)) = communicate(5).$$

If two actions do *not* communicate, e.g., continuing the example, when the sent and received values do not match, then their communication is not defined. For example,

$$\gamma(send(5), receive(6)) \text{ is not defined.}$$

Not every function is a meaningful communication function. For example, $\gamma(send(5), receive(5))$ and $\gamma(receive(5), send(5))$ are typically expected to give the same result. These ideas lead to the following definition of a communication function.

Definition 7.2.1 (Communication function) As always, set A is the set of atomic actions. A *communication function* on A is a partial, binary function $\gamma : A \times A \to A$ satisfying the following conditions:

(i) for all $a, b \in A$: $\gamma(a, b) = \gamma(b, a)$, i.e., communication is *commutative*;

(ii) for all $a, b, c \in A$: $\gamma(\gamma(a, b), c) = \gamma(a, \gamma(b, c))$, i.e., communication is *associative*.

https://doi.org/10.1017/CBO9781139195003.011 Published online by Cambridge University Press

Note that equations as above for partial functions imply that one side of the equation is defined exactly when the other side is. If $\gamma(a, b)$ for certain $a, b \in A$ is not defined, it is said that a and b *do not communicate*. An action $c \in A$ such that $c = \gamma(a, b)$ for certain a, b is called a *communication action*. A communication $\gamma(a, b)$ is called a *binary* communication.

A communication function in principle specifies communication between two actions, the binary communications. The result of such a communication can itself be allowed to communicate with other actions, which means that it is possible to specify communication among more than two actions. The associativity property in the definition of a communication function implies that the result of such communications is always the same, independent of the order in which the binary communications among pairs of actions are resolved. In fact, it follows from the commutativity and associativity requirements that it is possible to simplify expressions like $\gamma(a, \gamma(\gamma(b, c), d))$, and simply write $\gamma(a, b, c, d)$.

Having introduced the notion of a communication function, interaction between two components in a parallel composition can be described by the following deduction rule, where x, x', y, and y' are arbitrary process terms, γ is a communication function, and a, b, and c are actions:

$$\frac{x \xrightarrow{a} x' \quad y \xrightarrow{b} y' \quad \gamma(a, b) = c}{x \parallel y \xrightarrow{c} x' \parallel y'} .$$

Sometimes, one would like to express that communication is restricted. For example, it could be the case that only binary communications, i.e., communications involving only two actions, are possible. An action $\gamma(a, b, c)$ involves three actions, and is therefore called a *ternary* communication. It is an example of so-called higher-order communication. In many applications, higher-order communication is not possible and only binary communication occurs. This is called *handshaking* and can be expressed as follows.

Definition 7.2.2 (Handshaking communication) Let $\gamma : A \times A \to A$ be a communication function. If for all $a, b, c \in A$, $\gamma(a, b, c)$ is not defined, then γ is said to be a *handshaking* communication function.

In some applications, there is no communication whatsoever between the components. In such cases, no two atomic actions $a, b \in A$ communicate, i.e., $\gamma(a, b)$ is undefined for all $a, b \in A$. This is indicated by writing $\emptyset$ for γ, thus referring to the characterization of γ as the set $\{(a, b, c) \in A^3 \mid \gamma(a, b) = c\}$.

These two examples illustrate that the concept of a communication function is rather flexible. The deduction rule for communication does not change when

restricting the allowed communication functions beyond Definition 7.2.1 as in the above two examples. Note however, that the deduction rule for communication does not allow the derivation of any communication actions if $\gamma = \emptyset$, and is therefore redundant in that case, which is the expected result.

Finally, it is necessary to consider termination. Assume some parallel composition $x \parallel y$, with x and y process terms. Successful termination of the composition can be observed when both x and y are able to terminate. If one component still needs to execute actions, or cannot terminate successfully, then the composite process cannot terminate, and, for example, cannot pass control to a process following $x \parallel y$ (composed sequentially). This can be captured in the following deduction rule, with x and y arbitrary process terms:

$$\frac{x\downarrow \quad y\downarrow}{x \parallel y\downarrow} \ .$$

The next section presents a simple example illustrating the concepts introduced so far in this chapter. Section 7.4 then follows with the development of a basic equational theory for communicating processes.

Exercises

7.2.1 On the basis of the intuition given in this section, try to reason which of the following identities are correct. Assume there is no communication, i.e., $\gamma = \emptyset$.

(a) $a.1 \parallel b.1 = a.b.1 + b.a.1$.
(b) $a.b.1 \parallel c.1 = a.b.c.1 + a.c.b.1 + c.a.b.1$.
(c) $a.b.1 \parallel c.1 = a.(b.c.1 + c.b.1) + c.a.b.1$.
(d) $(a.1 + b.1) \parallel c.1 = a.c.1 + b.c.1 + c.(a.1 + b.1)$.
(e) $a.1 \parallel 0 = a.0$.
(f) $a.1 \parallel 0 = a.1$.

7.2.2 Formally describe the situation where only binary and ternary communication are possible, and none of higher order.

7.2.3 Is it possible to describe the situation where *only* ternary communication is possible, and not binary?

7.3 Standard communication

Many examples of communication involve the transmission of data at ports. It turns out to be useful to have some standardized notation for so-called *standard communication*.

https://doi.org/10.1017/CBO9781139195003.011 Published online by Cambridge University Press

Assume a number of *locations*, that are interconnected by *ports* or *channels*. Figure 7.1 gives an example. The system in the example has locations A, B, and C and ports ia, ab, ac, and co. Locations correspond to processes (represented by a closed term or a guarded recursive specification). Standard communication allows only binary communication, i.e., handshaking is assumed. Ports that connect two locations are called *internal*, ports attached to one location only are *external*. External ports can be used for communication between the system of processes and its environment. In the example of Figure 7.1, ports ab and ac are internal, and ia and co are external. The communication aspects of systems like those in Figure 7.1 can be captured via the following definition.

Definition 7.3.1 (Standard communication) Let D be a set of data elements; let P be a set of port names. Assume, for each port $p \in P$ and each $d \in D$, the presence of the following atomic actions in the set of actions A:

- $p!d$ (send data element d at port p);
- $p?d$ (receive d at p);
- $p⁉d$ (communicate d at p).

The *standard communication function* $\gamma_S : A \times A \to A$ is given by the following equation:

$$\gamma_S(p!d, p?d) = \gamma_S(p?d, p!d) = p⁉d,$$

for any $p \in P$ and $d \in D$; γ_S is not defined otherwise.

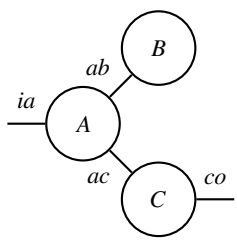

Fig. 7.1. A network of communicating processes.

Consider again the example of Figure 7.1. Using the actions predefined in Definition 7.3.1, process A might for example be specified by the term $ia?d.(ab!d.1 \parallel ac!d.1)$, for some data element d. Processes B and C might be defined as $ab?d.1$ and $ac?d.co!d.1$, respectively.

The system of Figure 7.1 as a whole can be described as a parallel composition of the processes involved:

https://doi.org/10.1017/CBO9781139195003.011 Published online by Cambridge University Press

$$ia?d.(ab!d.1 \parallel ac!d.1) \parallel ab?d.1 \parallel ac?d.co!d.1.$$

Following the intuition of the merge operator and its operational description discussed in the previous sections, this parallel composition allows arbitrary interleavings of the actions of the constituent processes, including communication actions of synchronizing processes. However, communication is not enforced. It could be desirable to enforce communication over the internal ports *ab* and *ac*. This can be done using the encapsulation operator of Section 6.7 by blocking isolated send and receive actions over these internal ports, see Exercise 7.3.1:

$$\partial_{\{p?d,p!d|p\in\{ab,ac\},d\in D\}}(ia?d.(ab!d.1 \parallel ac!d.1) \parallel ab?d.1 \parallel ac?d.co!d.1).$$

Exercises

7.3.1 Assume that a complete execution sequence of a system is a sequence of actions that can be performed by the system and that leads to successful termination. Provide three complete execution sequences of the example system discussed in this section when it is not assumed that any form of communication is enforced. Provide a complete execution sequence when assuming that communication over the internal ports *ab* and *ac* is enforced.

7.3.2 Provide a recursive specification for a stack with input port i and output port o using the actions of Definition 7.3.1 (Standard communication).

7.4 The process theory BCP

Coming up with a set of axioms that fully captures the intuition of communicating parallel processes given in the previous sections is no easy matter. In fact, Moller proved in (Moller, 1989) that no finite direct axiomatization exists. This means that there is no finite ground-complete axiomatization in the signature of the basic theory BSP(A) of Section 4.4 extended with the merge operator only of the standard term model based on bisimilarity obtained from the deduction rules of Section 7.2. In order to give a finite axiomatization nonetheless, an artifice is needed in the form of auxiliary operators.

First, consider the four deduction rules given in Section 7.2. The first rule says that $x \parallel y$ can start by executing a step from x, and the second rule says that $x \parallel y$ can start by executing a step from y. The last two rules concern joint activity. Either x and y execute an interaction, or they join in termination.

https://doi.org/10.1017/CBO9781139195003.011 Published online by Cambridge University Press

Thus, it can be seen that parallel composition is broken up into three alternatives, namely the part where the first step comes from x, the part where the first step comes from y, and the part where x and y execute together. Two new operators are introduced in order to express these options: $x \between y$ denotes the parallel composition of x and y with the restriction that the first step comes from x; $x \mid y$ denotes the parallel composition of x and y starting with a joint activity. Using this so-called *left-merge* operator $\between$ and *communication-merge* operator $\mid$, the alternatives of parallel composition can be presented as follows. It is assumed that $\between$ and $\mid$ have the same binding priority as $\parallel$, i.e., they bind stronger than choice and weaker than action prefix.

$$x \parallel y = x \between y + y \between x + x \mid y.$$

Implicitly, it is assumed here that the operators $\parallel$ and $\mid$ will be commutative, so that the order in which the components are presented is not relevant. Using the above equality, the problem of axiomatizing the $\parallel$ operator is replaced by the problem of axiomatizing the $\between$ and $\mid$ operators. Surprisingly, the latter problem is relatively straightforward.

First, the left-merge operator is considered. A case analysis following the structure of the left-hand argument provides the desired axioms. Let x, y, z be processes, and $a \in A$ an action.

First of all, the case of the prefix operators is not difficult:

$$a.x \between y = a.(x \parallel y).$$

In words, in a parallel composition of processes $a.x$ and y where the first step is from $a.x$, this first step must be an a. What remains is the parallel composition of x and y (without restriction).

Second, alternative composition is also straightforward.

$$(x + y) \between z = x \between z + y \between z.$$

It is crucial to notice that the moment of choice on both sides is the same: the choice is made by the execution of the first action. This is in contrast to the situation with the $\parallel$ operator where $(x + y) \parallel z$ is not equal to $x \parallel z + y \parallel z$, as the first step might be a step from z which means that the choice in the left argument of $(x + y) \parallel z$ is not decided whereas the choice in $x \parallel z + y \parallel z$ is decided.

Finally, what remains is the behavior of the left merge with respect to the termination constants 0 and 1. The termination behavior of a parallel composition is coded into the communication-merge operator, because termination and communication both require some form of synchronization, which means they fit naturally together. Therefore, termination behavior is of no concern for the left merge. The following axioms follow the intuition that a (successfully or

unsuccessfully) terminated process cannot perform a step, which implies that these constants as the left operand of a left merge lead to inaction.

$$0 \between x = 0 \quad \text{and} \quad 1 \between x = 0.$$

Next, the communication-merge operator is considered. A case analysis following the structure of the two arguments of the communication merge yields the axioms. Let x, y, z be processes, and $a, b, c \in A$ actions.

As the communication merge involves activity from both sides, it distributes over choice:

$$(x + y) \mid z = x \mid z + y \mid z \quad \text{and} \quad x \mid (y + z) = x \mid y + x \mid z.$$

Obviously, an inaction constant 0 on one side of a communication merge allows no joint activity whatsoever:

$$0 \mid x = 0 \quad \text{and} \quad x \mid 0 = 0.$$

What remains are the cases where both sides are action-prefix terms or empty processes 1. Successful termination, i.e., both sides of the communication merge are 1, is simple:

$$1 \mid 1 = 1.$$

If both sides are action prefixes, the result is based on the communication function γ. If γ is defined for the involved actions, then the communicating process can actually perform the defined communication action, and then proceed as an unconstrained parallel composition of the remaining behaviors of both operands of the communication merge; if γ is undefined, the communicating process cannot perform any action at all.

$$\begin{array}{ll} a.x \mid b.y = c.(x \parallel y) & \text{if } \gamma(a, b) = c \\ a.x \mid b.y = 0 & \text{if } \gamma(a, b) \text{ is not defined.} \end{array}$$

Finally, communication-merge expressions combining an action prefix and an empty process lead to inaction, because both a joint action and synchronized successful termination are impossible:

$$a.x \mid 1 = 0 \quad \text{and} \quad 1 \mid a.x = 0.$$

Note that the axiom system presented below contains an axiom stipulating the commutativity of the communication merge. This allows to save on the number of axioms required for the communication-merge operator. From the three cases with symmetric axioms discussed above, only one axiom for each case is needed. Assuming commutativity of the communication merge does not influence the equalities that can be derived for closed terms in the theory presented below. However, it does allow the derivation of equalities between open terms that cannot be derived from the theory without this axiom (and containing all six axioms for the above three symmetric cases).

https://doi.org/10.1017/CBO9781139195003.011 Published online by Cambridge University Press

The formal definition of process theory BCP, the theory of Basic Communicating Processes, is given in Table 7.1. It extends theory $(\mathrm{BSP} + \mathrm{DH})(A)$ of Table 6.8. The theory thus includes the encapsulation operator because this operator is essential in describing communication between processes, as illustrated by the example in Section 7.3. As before, the theory has as a parameter the set of actions A. Besides this, it has as a second parameter a communication function $\gamma : A \times A \to A$ satisfying the conditions of Definition 7.2.1 (Communication function). The signature of process theory $\mathrm{BCP}(A, \gamma)$ extends the signature of the process theory $(\mathrm{BSP} + \mathrm{DH})(A)$ with the merge operator $\parallel$, the left-merge operator $\lfloor\!\lfloor$ and the communication-merge operator $\mid$. The three new operators bind stronger than choice but weaker than action prefix. The axioms of $\mathrm{BCP}(A, \gamma)$ are the axioms in Table 7.1 added to the axioms of theory $(\mathrm{BSP} + \mathrm{DH})(A)$ of Table 6.8. Note that Axioms LM3 and CM4–6 are actually *axiom schemes*: there is such an axiom for each combination of atomic actions $a, b, c \in A$ occurring in them.

$\mathrm{BCP}(A, \gamma)$

$(\mathrm{BSP} + \mathrm{DH})(A)$;

binary: $\_ \parallel \_$, $\_ \lfloor\!\lfloor \_$, $\_ \mid \_$;

x, y, z;

$x \parallel y = x \lfloor\!\lfloor y + y \lfloor\!\lfloor x + x \mid y$	M		$x \mid y = y \mid x$	SC1
$0 \lfloor\!\lfloor x = 0$	LM1			
$1 \lfloor\!\lfloor x = 0$	LM2		$x \parallel 1 = x$	SC2
$a.x \lfloor\!\lfloor y = a.(x \parallel y)$	LM3		$1 \mid x + 1 = 1$	SC3
$(x + y) \lfloor\!\lfloor z = x \lfloor\!\lfloor z + y \lfloor\!\lfloor z$	LM4			
$0 \mid x = 0$	CM1		$(x \parallel y) \parallel z = x \parallel (y \parallel z)$	SC4
$(x + y) \mid z = x \mid z + y \mid z$	CM2		$(x \mid y) \mid z = x \mid (y \mid z)$	SC5
$1 \mid 1 = 1$	CM3		$(x \lfloor\!\lfloor y) \lfloor\!\lfloor z = x \lfloor\!\lfloor (y \parallel z)$	SC6
$a.x \mid 1 = 0$	CM4		$(x \mid y) \lfloor\!\lfloor z = x \mid (y \lfloor\!\lfloor z)$	SC7
$a.x \mid b.y = c.(x \parallel y)$	if $\gamma(a, b) = c$		CM5	
$a.x \mid b.y = 0$	if $\gamma(a, b)$ is not defined		CM6	

Table 7.1. The process theory $\mathrm{BCP}(A, \gamma)$ (with $a, b, c \in A$).

Axioms M, LM1–4, and CM1–6 of $\mathrm{BCP}(A, \gamma)$ have been discussed above. They serve to rewrite each closed term involving parallel composition operators into a closed $\mathrm{BSP}(A)$-term (see the elimination theorem below). Besides these axioms, the theory $\mathrm{BCP}(A, \gamma)$ contains seven additional axioms denoting properties of parallel composition, the so-called *Axioms of Standard Concurrency*. These properties are often convenient.

The commutativity and associativity of the merge operator, and the fact that

1 is an identity element, are the most important properties. Commutativity of the merge follows from SC1 and the commutativity of the choice operator (see Exercise 7.4.3). Associativity is posed as Axiom SC4. The fact that 1 is an identity element is stated by SC2. SC3 captures the fact that a communication with the empty process either results in inaction or successful termination and that no actions are possible, see Proposition 7.4.2 (Communication with the empty process) below. This axiom illustrates that the communication merge is used to capture the termination options of a parallel composition. The other axioms of standard concurrency are variants of SC4 involving the left-merge and communication-merge operators.

Example 7.4.1 (Communication) Consider a relay race with two runners, Alice and Bob. First, Alice runs some distance, then she passes the baton to Bob. Bob first takes the baton, and next runs some more. The processes executed can be given via closed BCP(A, γ)-terms as follows:

$$A = runA.give.1 \quad \text{and} \quad B = take.runB.1.$$

The only defined communication is $\gamma(give, take) = pass$ (and its commutative variant, of course). Applying the axioms of BCP(A, γ) gives the following result:

$$\begin{aligned}
&\mathrm{BCP}(A,\gamma)\vdash\\
&\quad A \parallel B = runA.(give.take.runB.1 + pass.runB.1\\
&\qquad\qquad + take.(give.runB.1 + runB.give.1)\\
&\qquad\qquad) +\\
&\qquad take.\ (runB.runA.give.1\\
&\qquad\qquad + runA.(give.runB.1 + runB.give.1)\\
&\qquad\qquad).
\end{aligned}$$

Obviously, many unwanted action sequences occur in this term, where 'halves' of communication actions occur by themselves. Communication can be enforced by the encapsulation operator, as already illustrated in Section 7.3. Defining $H = \{give, take\}$, it can be shown that

$$\mathrm{BCP}(A, \gamma) \vdash \partial_H(A \parallel B) = runA.pass.runB.1.$$

Thus, by blocking the separate components of communication actions, only the successful communications remain. This use of the encapsulation operator in fact explains its name: the operator prevents actions within its scope to communicate with actions outside.

Axiom SC3 of BCP(A, γ) may need some clarification. Recall the notion of summands introduced in Exercise 4.2.3. The following can be derived, for any process term x: $\mathrm{BCP}(A, \gamma) \vdash x = x \parallel 1 = x \between 1 + 1 \between x + x \mid 1 = x \between 1 +$

https://doi.org/10.1017/CBO9781139195003.011 Published online by Cambridge University Press

$1 \mid x$. In other words, any process x breaks down into two parts, namely $x \between 1$, being all of x except a possibly occurring 1 summand (see Axioms LM1–4, in particular LM2), and $1 \mid x$, capturing this possibly occurring 1 summand. As a consequence, $1 \mid x$ is either equal to 1 (if x has a 1 summand) or 0 (otherwise). This is what Axiom SC3 expresses: $1 \mid x$ is a summand of 1, so it is either 1 or 0. Formally, the following can be derived.

Proposition 7.4.2 (Communication with the empty process) For any term x, if $\mathrm{BCP}(A, \gamma) \vdash x = x + 1$, then $\mathrm{BCP}(A, \gamma) \vdash 1 \mid x = 1$; if $\mathrm{BCP}(A, \gamma) \vdash x = x \between 1$, then $\mathrm{BCP}(A, \gamma) \vdash 1 \mid x = 0$.

Proof The two properties can be proven via straightforward equational reasoning, see Exercise 7.4.5. □

Thus, when a process contains a 1 summand, it can be derived from the axioms of $\mathrm{BCP}(A, \gamma)$ that $\mathrm{BCP}(A, \gamma) \vdash 1 \mid x = 1$. However, the general result that $1 \mid x$ is equal to 0 if x does not have a 1 summand cannot be derived from the axioms of $\mathrm{BCP}(A, \gamma)$ for general open $\mathrm{BCP}(A, \gamma)$-terms. It is only possible to derive the above conditional property (the second property in Proposition 7.4.2) and an instance of it for guarded terms (see Proposition 7.6.3, in Section 7.6 which covers recursion).

The next theorem shows that the newly introduced operators can be eliminated from closed $\mathrm{BCP}(A, \gamma)$-terms. This means that the introduction of parallel composition does not increase expressiveness; the same set of processes can be specified as in the theory $\mathrm{BSP}(A)$, but in more different, often more intuitive and more compact, ways. In fact, the number of symbols in a term can increase exponentially when rewriting a $\mathrm{BCP}(A, \gamma)$-term to an equivalent $\mathrm{BSP}(A)$-term, see Exercise 7.4.11.

Theorem 7.4.3 (Elimination) For any closed $\mathrm{BCP}(A, \gamma)$-term p, there exists a closed $\mathrm{BSP}(A)$-term q such that $\mathrm{BCP}(A, \gamma) \vdash p = q$.

Proof Axioms M, LM1–4, and CM1–6 in Table 7.1 can be ordered from left to right as a term rewriting system. It is necessary to add rewrite rules corresponding to CM1, CM2, and CM4 where the arguments of the communication merge are swapped. This is harmless due to Axiom SC1. Furthermore, rewrite rules corresponding to the axioms of encapsulation (see Table 6.8) need to be added. Then, this proof follows the same pattern as earlier proofs of elimination theorems. Details are left for Exercise 7.4.6. □

The Axioms of Standard Concurrency SC1 through SC7 of $\mathrm{BCP}(A, \gamma)$ are

https://doi.org/10.1017/CBO9781139195003.011 Published online by Cambridge University Press

basic axioms of the theory of parallel processes, and they cannot be derived from the other axioms. However, except for SC1, they can be derived for all closed terms. (If the counterparts of CM1, CM2, and CM4 are added to the theory as in the proof of Theorem 7.4.3, then also SC1 can be derived for closed terms.)

Theorem 7.4.4 (Standard concurrency) For closed BCP(A, γ)-terms, the Axioms of Standard Concurrency SC2–7 are derivable from the other axioms of BCP(A, γ).

Proof The property for Axiom SC2 is proven by structural induction. In view of Theorem 7.4.3 (Elimination), assume that p is a closed BSP(A)-term.

(i) Assume $p \equiv 0$. Consequently, BCP$(A, \gamma) \vdash p \parallel 1 = 0 \lfloor\!\lfloor 1 + 1 \lfloor\!\lfloor 0 + 0 \mid 1 = 0 + 0 + 0 = 0 = p$.
(ii) Assume $p \equiv 1$. BCP$(A, \gamma) \vdash p \parallel 1 = 1 \lfloor\!\lfloor 1 + 1 \lfloor\!\lfloor 1 + 1 \mid 1 = 0 + 0 + 1 = 1 = p$.
(iii) Assume $p \equiv a.q$, for some $a \in A$ and closed BSP(A)-term q. It follows that BCP$(A, \gamma) \vdash p \parallel 1 = a.q \lfloor\!\lfloor 1 + 1 \lfloor\!\lfloor a.q + a.q \mid 1 = a.(q \parallel 1) + 0 + 0 = a.q = p$.
(iv) Assume $p \equiv p_1 + p_2$, for some closed BSP(A)-terms p_1 and p_2. Then, BCP$(A, \gamma) \vdash p \parallel 1 = p \lfloor\!\lfloor 1 + 1 \lfloor\!\lfloor p + p \mid 1 = p_1 \lfloor\!\lfloor 1 + p_2 \lfloor\!\lfloor 1 + 0 + p_1 \mid 1 + p_2 \mid 1 = p_1 \lfloor\!\lfloor 1 + p_2 \lfloor\!\lfloor 1 + 1 \lfloor\!\lfloor p_1 + 1 \lfloor\!\lfloor p_2 + p_1 \mid 1 + p_2 \mid 1 = p_1 \parallel 1 + p_2 \parallel 1 = p_1 + p_2 = p$.

The proof for Axiom SC3 is equally straightforward. Assume again that p is a closed BSP(A)-term.

(i) Assume $p \equiv 0$. Consequently, BCP$(A, \gamma) \vdash 1 \mid p + 1 = 1 \mid 0 + 1 = 0 \mid 1 + 1 = 0 + 1 = 1$.
(ii) Assume $p \equiv 1$. BCP$(A, \gamma) \vdash 1 \mid p + 1 = 1 \mid 1 + 1 = 1 + 1 = 1$.
(iii) Assume $p \equiv a.q$, for some action $a \in A$ and closed BSP(A)-term q. BCP$(A, \gamma) \vdash 1 \mid p + 1 = 1 \mid a.q + 1 = a.q \mid 1 + 1 = 0 + 1 = 1$.
(iv) Assume $p \equiv p_1 + p_2$, for some closed BSP(A)-terms p_1 and p_2. It then follows that BCP$(A, \gamma) \vdash 1 \mid p + 1 = 1 \mid (p_1 + p_2) + 1 = 1 \mid p_1 + 1 \mid p_2 + 1 = 1 \mid p_1 + 1 + 1 \mid p_2 + 1 = 1 + 1 = 1$.

For the other four axioms, the theorem is proved simultaneously by natural induction on the total number of symbols in closed BCP(A, γ)-terms p, q, and r. Assume that this number of symbols is k. Based on Theorem 7.4.3 (Elimination), assume again that p, q, and r are BSP(A)-terms.
The base case of the proof, where the number of symbols is three, and thus

https://doi.org/10.1017/CBO9781139195003.011 Published online by Cambridge University Press

all three terms are 1 or 0, is left to the reader. In the induction step, it can be assumed that all four equalities hold for all triples of closed terms containing in total fewer than k symbols, for some natural number k. As p is a closed BSP(A)-term, it follows from Proposition 5.5.25 (HNF property) that it may be assumed that p can be written as follows:

$$p \equiv \sum_{i<n} a_i.p_i \ (+1),$$

for a certain natural number $n \geq 0$, atomic actions a_i and closed terms p_i with fewer symbols than p, and a 1 summand which may or may not be present.
For Axiom SC6, consider the following derivation, using induction (with respect to Axiom SC4) in the fifth step:

$$\begin{aligned}
\mathrm{BCP}(A,\gamma) \vdash (p \between q) \between r &= \left(\left(\sum_{i<n} a_i.p_i \ (+1)\right) \between q\right) \between r \\
&= \left(\sum_{i<n} a_i.p_i \between q \ (+1 \between q)\right) \between r \\
&= \sum_{i<n} a_i.(p_i \parallel q) \between r \\
&= \sum_{i<n} a_i.((p_i \parallel q) \parallel r) \\
&= \sum_{i<n} a_i.(p_i \parallel (q \parallel r)) \\
&= \sum_{i<n} a_i.p_i \between (q \parallel r) \\
&= \sum_{i<n} a_i.p_i \between (q \parallel r)(+1 \between (q \parallel r)) \\
&= (\sum_{i<n} a_i.p_i(+1)) \between (q \parallel r) \\
&= p \between (q \parallel r).
\end{aligned}$$

For Axiom SC7, besides p also q is written in sum notation:

$$q \equiv \sum_{j<m} b_j.q_j \ (+1),$$

for natural number $m \geq 0$, actions $b_j \in A$, and simpler closed terms q_j. The following derivation uses the induction hypothesis in the sixth step. Note that a communication merge has only a termination option (a 1 summand) if both its arguments have a termination option. Therefore, in the one-but-last step of the derivation, the optional 1 summands can be added because the resulting right argument of the communication-merge operator does not have a termination option due to the occurrence of the left-merge operator.

$$\begin{aligned}
\mathrm{BCP}(A,\gamma) \vdash\ & (p \mid q) \between r \\
&= \left((\sum_{i<n} a_i.p_i \ (+1)) \mid (\sum_{j<m} b_j.q_j \ (+1))\right) \between r
\end{aligned}$$

https://doi.org/10.1017/CBO9781139195003.011 Published online by Cambridge University Press

$$
\begin{aligned}
&= \left(\sum_{i<n}\sum_{j<m} a_i.p_i \mid b_j.q_j\right) \between r \ (+1 \between r)\\
&= \left(\sum_{i,\,j \text{ with } \gamma(a_i,b_j) \text{ defined}} \gamma(a_i,b_j).(p_i \parallel q_j)\right) \between r\\
&= \sum_{i,\,j \text{ with } \gamma(a_i,b_j) \text{ defined}} \gamma(a_i,b_j).(p_i \parallel q_j) \between r\\
&= \sum_{i,\,j \text{ with } \gamma(a_i,b_j) \text{ defined}} \gamma(a_i,b_j).((p_i \parallel q_j) \parallel r)\\
&= \sum_{i,\,j \text{ with } \gamma(a_i,b_j) \text{ defined}} \gamma(a_i,b_j).(p_i \parallel (q_j \parallel r))\\
&= (\sum_{i<n} a_i.p_i) \mid (\sum_{j<m} b_j.(q_j \parallel r))\\
&= (\sum_{i<n} a_i.p_i) \mid (\sum_{j<m} b_j.q_j \between r)\\
&= (\sum_{i<n} a_i.p_i(+1)) \mid (\sum_{j<m} (b_j.q_j(+1)) \between r)\\
&= p \mid (q \between r).
\end{aligned}
$$

For Axiom SC5, all three terms are written in sum notation, so in addition to the earlier assumptions,

$$r \equiv \sum_{k<l} c_k.r_k \ (+1),$$

for natural number $l \geq 0$, $c_k \in A$, and simpler closed terms r_k. Notice that terms of the form $(p \mid q) \mid r$ and $p \mid (q \mid r)$ only contain a 1 summand if all three of p, q, and r contain a 1 summand. The following derivation uses in the fourth step the induction hypothesis and the associativity of the communication function.

$$
\begin{aligned}
\mathrm{BCP}(A,\gamma) \vdash\ & (p \mid q) \mid r\\
&= \left(\sum_{i<n} a_i.p_i \ (+1) \mid \sum_{j<m} b_j.q_j \ (+1)\right) \mid r\\
&= \left(\sum_{i,\,j \text{ with } \gamma(a_i,b_j) \text{ defined}} \gamma(a_i,b_j).(p_i \parallel q_j) \ (+1)\right) \mid r\\
&= \sum_{i,\,j,\,k \text{ with } \gamma(a_i,b_j,c_k) \text{ defined}} \gamma(\gamma(a_i,b_j),c_k).((p_i \parallel q_j) \parallel r_k) \ (+1)\\
&= \sum_{i,\,j,\,k \text{ with } \gamma(a_i,b_j,c_k) \text{ defined}} \gamma(a_i,\gamma(b_j,c_k)).(p_i \parallel (q_j \parallel r_k)) \ (+1)\\
&= (\sum_{i<n} a_i.p_i(+1)) \mid \left(\sum_{j,\,k \text{ with } \gamma(b_j,c_k) \text{ defined}} \gamma(b_j,c_k).(q_j \parallel r_k) \ (+1)\right)\\
&= p \mid (q \mid r).
\end{aligned}
$$

Thus, SC5–7 are proved for all triples of closed terms with total number of symbols k. This fact is now used in the following derivation, proving the theorem for SC4 for all triples of closed terms with total number of symbols k:

https://doi.org/10.1017/CBO9781139195003.011 Published online by Cambridge University Press

$$
\begin{aligned}
\mathrm{BCP}(A,\gamma) \vdash\ & (p \parallel q) \parallel r \\
&= (p \parallel q) \between r + r \between (p \parallel q) + (p \parallel q) \mid r \\
&= (p \between q + q \between p + p \mid q) \between r + r \between (p \parallel q) + r \mid (p \parallel q) \\
&= (p \between q) \between r + (q \between p) \between r + (p \mid q) \between r + r \between (p \parallel q) \\
&\qquad + r \mid (p \between q) + r \mid (q \between p) + r \mid (p \mid q) \\
&= p \between (q \parallel r) + q \between (p \parallel r) + p \mid (q \between r) + r \between (q \parallel p) \\
&\qquad + (r \mid p) \between q + (r \mid q) \between p + (p \mid q) \mid r \\
&= p \between (q \parallel r) + q \between (r \parallel p) + p \mid (q \between r) + (r \between q) \between p \\
&\qquad + (p \mid r) \between q + (q \mid r) \between p + p \mid (q \mid r) \\
&= p \between (q \parallel r) + (q \between r) \between p + p \mid (q \between r) + (r \between q) \between p \\
&\qquad + p \mid (r \between q) + (q \mid r) \between p + p \mid (q \mid r) \\
&= p \between (q \parallel r) + (q \parallel r) \between p + p \mid (q \parallel r) \\
&= p \parallel (q \parallel r).
\end{aligned}
$$

Induction then proves the theorem for SC4–7 for all closed $\mathrm{BCP}(A,\gamma)$-terms. □

Because of Axiom SC4, it is not necessary to write parentheses in expressions like $x \parallel y \parallel z$. Therefore, a notation for generalized parallel composition is introduced, similar to Notation 5.5.19 (Generalized choice). Considering Axiom SC2, the parallel composition of an empty set of process terms results in the empty process.

Notation 7.4.5 (Generalized merge) Let $I \subset \mathbf{N}$ be some finite index set of natural numbers, $j \in \mathbf{N} \setminus I$ a fresh index not in I, and t_i, for all $i \in I \cup \{j\}$, arbitrary terms in some process theory containing the merge operator $\parallel$.

$$\parallel_{i\in\emptyset} t_i \equiv 1 \text{ and } \parallel_{i\in I\cup\{j\}} t_i \equiv t_j \parallel \parallel_{i\in I} t_i.$$

Similarly, based on Axiom SC5, a generalized notation for the communication merge can be introduced, where it is important to observe that the communication merge does not have an identity element (which implies that the notation cannot be defined for an empty index set).

Notation 7.4.6 (Generalized communication merge) Let $I \subset \mathbf{N}$ be some finite, non-empty index set of natural numbers, $j \in \mathbf{N} \setminus I$ a fresh index not in I, and t_i, for all $i \in I \cup \{j\}$, arbitrary terms in some process theory containing the communication merge operator $\mid$.

$$\mid_{i\in\{j\}} t_i \equiv t_j \text{ and } \mid_{i\in I\cup\{j\}} t_i \equiv t_j \mid \mid_{i\in I} t_i.$$

Using these notations, the following expansion theorem can be formulated.

https://doi.org/10.1017/CBO9781139195003.011 Published online by Cambridge University Press

This theorem is the main tool in breaking down the parallel composition of a number of processes, and it is often used in the remainder.

Theorem 7.4.7 (Expansion theorem) Let $I \subset \mathbf{N}$ be some finite, non-empty index set of natural numbers, and t_i, for all $i \in I$, arbitrary BCP(A, γ)-terms.

$$\mathrm{BCP}(A,\gamma) \vdash \parallel_{i\in I} t_i = \sum_{\emptyset\neq J\subset I} \left(\mid_{j\in J} t_j\right) \between \left(\parallel_{i\in I\setminus J} t_i\right) + \mid_{i\in I} t_i .$$

Proof The proof is by induction on the cardinality of index set I. In the base case where the cardinality is 1, say the index set is singleton $\{0\}$, the theorem reduces to a trivial equality: $\mathrm{BCP}(A,\gamma) \vdash t_0 \parallel 1 = t_0 = 0 + t_0$. For the induction step, assume that the theorem is proven for all index sets of cardinality k, with $k \geq 1$, and assume that index set I of cardinality $k+1$ is equal to $K \cup \{k\}$ with $K = \{0, 1, \ldots, k-1\}$. The following derivation inductively applies the expansion theorem in the third step. The first four summands in the expression after the fourth step correspond to the cases where the set of indices (1) is a non-empty, strict subset of K, (2) equals K, (3) equals singleton $\{k\}$, and (4) contains index k and some non-empty, strict subset of K. Together, these four cases cover all non-empty, strict subsets of $I (= K \cup \{k\})$. These observations clarify the last two steps of the derivation.

$$\begin{aligned}
&\mathrm{BCP}(A,\gamma) \vdash \\
&\quad \parallel_{i\in I} t_i = \left(\parallel_{i\in K} t_i\right) \parallel t_k \\
&\qquad = \left(\parallel_{i\in K} t_i\right) \between t_k + t_k \between \left(\parallel_{i\in K} t_i\right) + \left(\parallel_{i\in K} t_i\right) \mid t_k \\
&\qquad = \left(\sum_{\emptyset\neq J\subset K} \left(\mid_{j\in J} t_j\right) \between \left(\parallel_{i\in K\setminus J} t_i\right) + \mid_{i\in K} t_i\right) \between t_k \\
&\qquad\quad + t_k \between \left(\parallel_{i\in K} t_i\right) \\
&\qquad\quad + \left(\sum_{\emptyset\neq J\subset K} \left(\mid_{j\in J} t_j\right) \between \left(\parallel_{i\in K\setminus J} t_i\right) + \mid_{i\in K} t_i\right) \mid t_k \\
&\qquad = \sum_{\emptyset\neq J\subset K} \left(\mid_{j\in J} t_j\right) \between \left(\left(\parallel_{i\in K\setminus J} t_i\right) \parallel t_k\right) \\
&\qquad\quad + \left(\mid_{i\in K} t_i\right) \between t_k \\
&\qquad\quad + t_k \between \left(\parallel_{i\in K} t_i\right) \\
&\qquad\quad + \sum_{\emptyset\neq J\subset K} \left(\left(\mid_{j\in J} t_j\right) \mid t_k\right) \between \left(\parallel_{i\in K\setminus J} t_i\right) \\
&\qquad\quad + \left(\mid_{i\in K} t_i\right) \mid t_k
\end{aligned}$$

https://doi.org/10.1017/CBO9781139195003.011 Published online by Cambridge University Press

$$
\begin{aligned}
&= \sum_{\emptyset \neq J \subset K} \Big(\Big| _{j\in J} t_j \Big) \between \Big(\Big\| _{i\in I\setminus J} t_i \Big) \\
&\quad + \sum_{J=K} \Big(\Big| _{j\in J} t_j \Big) \between \Big(\Big\| _{i\in I\setminus J} t_i \Big) \\
&\quad + \sum_{J=\{k\}} \Big(\Big| _{j\in J} t_j \Big) \between \Big(\Big\| _{i\in I\setminus J} t_i \Big) \\
&\quad + \sum_{\{k\}\subset J\subset I} \Big(\Big| _{j\in J} t_j \Big) \between \Big(\Big\| _{i\in I\setminus J} t_i \Big) \\
&\quad + \Big| _{i\in I} t_i \\
&= \sum_{\emptyset \neq J \subset I} \Big(\Big| _{j\in J} t_j \Big) \between \Big(\Big\| _{i\in I\setminus J} t_i \Big) + \Big| _{i\in I} t_i .
\end{aligned}
$$

The desired result now follows by induction. □

The expansion theorem is an important means to rewrite a parallel composition into sequential behaviors. Therefore, it is important to understand what it says. In the proof of the theorem, it was already noted that for an index set of size one, it is a trivial statement. For an index set of cardinality two, the expansion theorem reduces exactly to Axiom M of theory BCP(A, γ). For an index set of size three, it can be written out as follows:

$$
\begin{aligned}
&\mathrm{BCP}(A,\gamma) \vdash \\
&\quad x \parallel y \parallel z = x \between (y \parallel z) + y \between (x \parallel z) + z \between (y \parallel z) \\
&\qquad\qquad + (x \mid y) \between z + (x \mid z) \between y + (y \mid z) \between x \\
&\qquad\qquad + x \mid y \mid z .
\end{aligned}
$$

In words, in a parallel composition of three processes, the first step is a step from one of the three processes (the first three summands of the right-hand term) or a communication step between two of the three processes (the next three summands) or a synchronization between all processes (the last summand).

The remainder of this section briefly considers two special cases of parallel processes, the case when communication is absent and the case when only handshaking communication (see Definition 7.2.2 (Handshaking communication)) is allowed. These special cases lead to additional axioms in the equational theory to capture the appropriate restrictions on communication, and to specialized versions of the expansion theorem.

First, consider the case without interaction between processes. This can be captured by assuming that the communication function is nowhere defined, i.e., $\gamma = \emptyset$. Parallel composition without interaction between processes is often called a *free merge*. In this case, Axiom SC3 of theory BCP(A, γ) can be strengthened to the *Free-Merge Axiom*, given in Table 7.2. The axiom states that the synchronization of two processes either results in inaction (0) or in successful termination (1). Thus, communication actions are indeed no longer

https://doi.org/10.1017/CBO9781139195003.011 Published online by Cambridge University Press

possible. The role of the communication merge in the resulting equational theory reduces to capturing the termination options of parallel processes.

$$\frac{}{(\mathrm{BCP}+\mathrm{FMA})(A,\emptyset)\ \mathrm{BCP}(A,\emptyset);}$$

$$\frac{}{x,y;}$$

$$x \mid y + 1 = 1 \qquad \mathrm{FMA}$$

Table 7.2. Basic communicating processes with a free merge.

For closed terms, the Free-Merge Axiom is derivable from the axioms of the equational theory $\mathrm{BCP}(A,\emptyset)$.

Proposition 7.4.8 (Free-Merge Axiom) For closed $\mathrm{BCP}(A,\emptyset)$-terms p and q, $\mathrm{BCP}(A,\emptyset) \vdash p \mid q + 1 = 1$.

Proof Exercise 7.4.7. □

In the presence of the Free-Merge Axiom, the expansion theorem of Theorem 7.4.7 can be simplified. All terms containing both a communication merge and a left merge can be omitted, because they result in inaction. The summand in the expansion corresponding to the communication merge of all terms in the parallel composition only denotes a possible termination option, i.e., this summand is either 1 or 0.

Theorem 7.4.9 (Expansion theorem for free merge) Let $I \subset \mathbf{N}$ be some finite, non-empty index set of natural numbers, and t_i, for all $i \in I$, arbitrary $(\mathrm{BCP}+\mathrm{FMA})(A,\emptyset)$-terms.

$$(\mathrm{BCP}+\mathrm{FMA})(A,\emptyset) \vdash \mathop{\Vert}_{i\in I} t_i = \sum_{i\in I} t_i \mathbin{\lfloor\!\lfloor} \Big(\mathop{\Vert}_{j\in I\setminus\{i\}} t_j\Big) + \mathop{\mid}_{i\in I} t_i .$$

Proof The desired result follows directly from the general expansion theorem, Theorem 7.4.7, by observing that Axiom FMA implies that $(\mathrm{BCP}+\mathrm{FMA})(A,\emptyset) \vdash (x \mid y) \mathbin{\lfloor\!\lfloor} z = 0$ for arbitrary terms x, y, and z (Exercise 7.4.8). □

The second special case that is considered is the case where there is only binary communication, called handshaking before, i.e., the communication function γ satisfies that $\gamma(a,b,c)$ is not defined, for any atomic actions a, b, c (see Definition 7.2.2 (Handshaking communication)). This case can also be formulated in terms of an extra axiom, called the *Handshaking Axiom*, given

https://doi.org/10.1017/CBO9781139195003.011 Published online by Cambridge University Press

in Table 7.3. The axiom states that any communication involving three (or more) processes results either in inaction or in successful termination.

$$\frac{}{(\mathrm{BCP}+\mathrm{HA})(A,\gamma)}$$
$$\frac{}{\mathrm{BCP}(A,\gamma);}$$
$$\frac{}{-}$$
$$\frac{}{x, y, z;}$$
$$x \mid y \mid z + 1 = 1 \qquad \mathrm{HA}$$

Table 7.3. Basic communicating processes with handshaking communication (with γ a handshaking communication function).

The Handshaking Axiom is derivable for all closed terms when assuming a handshaking communication function. Furthermore, the following expansion theorem is obtained.

Theorem 7.4.10 (Expansion theorem, handshaking communication) Let $I \subset \mathbf{N}$ be some finite, non-empty index set of natural numbers, and t_i, for all $i \in I$, arbitrary $(\mathrm{BCP}+\mathrm{HA})(A,\gamma)$-terms.

$$(\mathrm{BCP}+\mathrm{HA})(A,\gamma) \vdash \between_{i\in I} t_i = \sum_{i\in I} t_i \between (\between_{j\in I\setminus\{i\}} t_j) + \sum_{i,j\in I, i\neq j} (t_i \mid t_j) \between (\between_{k\in I\setminus\{i,j\}} t_k) + \mid_{i\in I} t_i.$$

Proof The result follows from the expansion theorem of Theorem 7.4.7 and the observation that Axiom HA implies that $(\mathrm{BCP}+\mathrm{HA})(A,\gamma) \vdash (x \mid y \mid z) \between w = 0$ for terms x, y, z, and w. □

Exercises

7.4.1 Use the axioms of $\mathrm{BCP}(A,\gamma)$ to write the following terms without parallelism and encapsulation operators (i.e., eliminate $\between$, $\mid$ and ∂_H). The only defined communication is $\gamma(a,b) = c$ and $H = \{a,b\}$.

(a) $a.a.1 \between b.b.1$;
(b) $\partial_H(a.a.1 \between b.b.1)$;
(c) $(a.1 + b.1) \between (a.1 + b.1)$;
(d) $\partial_H((a.1 + b.1) \between (a.1 + b.1))$;
(e) $a.a.a.1 \between a.a.a.1$.

7.4.2 Let $\gamma(a,b) = c$ be the only defined communication and let $H = \{a,b\}$. Show that $\partial_H(c.(a.1 + b.1) \between b.1)$ is deadlock free, where

https://doi.org/10.1017/CBO9781139195003.011 Published online by Cambridge University Press

a closed BCP(A, γ)-term is deadlock free if and only if it is derivably equal to a closed BSP(A)-term without 0 occurrence. Show that $\partial_H((c.a.1 + c.b.1) \parallel b.1)$ does have a deadlock. This again illustrates the difference between terms like $c.a.1 + c.b.1$ and $c.(a.1 + b.1)$.

7.4.3 Prove that $\mathrm{BCP}(A, \gamma) \vdash x \parallel y = y \parallel x$.

7.4.4 Prove that, for each atomic action $a \in A$ and natural numbers $m, n \in \mathbf{N}$, $\mathrm{BCP}(A, \gamma) \vdash a^m 1 \parallel a^n 1 = a^{m+n} 1$, where a^k for any $k \in \mathbf{N}$ is the k-fold action prefix of Notation 4.6.6.

7.4.5 Prove Proposition 7.4.2 (Communication with the empty process).

7.4.6 Complete the proof of Theorem 7.4.3 (Elimination).

7.4.7 Prove Proposition 7.4.8 (Free-Merge Axiom).

7.4.8 Prove that the following identities follow from $(\mathrm{BCP} + \mathrm{FMA})(A, \emptyset)$, for arbitrary terms x, y, and z:

(a) $(x \mid y) \between z = 0$;
(b) if $x = x + 1$ and $y = y + 1$, then $x \mid y = 1$;
(c) if $x = x \between 1$ or $y = y \between 1$, then $x \mid y = 0$;
(d) $1 \mid x = 1 \mid x + 1 \mid x \mid y$.

7.4.9 In $(\mathrm{BCP} + \mathrm{FMA})(A, \emptyset)$, prove the following identities for all closed terms p and q, and action a.

(a) $a.p \mid q = 0$;
(b) $1 \mid p = 1 \mid p \mid p$;
(c) $p + p \mid q = p$;
(d) $p \mid p \mid q = p \mid q$;
(e) $p + p \parallel (1 \mid q) + p \parallel 0 = p + p \parallel 0$.

7.4.10 Write out the proofs of Theorems 7.4.9 (Expansion theorem for free merge) and 7.4.10 (Expansion theorem for handshaking communication) in full detail.

7.4.11 Consider Theorem 7.4.3 (Elimination). Argue that the elimination of parallelism operators from a closed BCP(A, γ)-term may result in a BSP(A)-term with a length that is exponential in the number of symbols of the original term. Write a computer program that implements the rewriting system used in the proof of Theorem 7.4.3. Use this program to eliminate the parallelism operators from term $a^3 1 \parallel b^3 1 \parallel c^3 1$, where d^n for action d and natural number n is the n-fold action prefix of Notation 4.6.6. For simplicity, assume there is no communication (take $\gamma = \emptyset$).

https://doi.org/10.1017/CBO9781139195003.011 Published online by Cambridge University Press

7.4.12 Give an axiomatization of the theory BCP$(A, \emptyset)$ without communication-merge operator, where the termination behavior is coded into the left-merge operator.

7.5 The term model

In the previous section, the process theory BCP(A, γ), with A a set of actions and γ a communication function on A, has been introduced. This section considers a model of this equational theory, using a set of operational rules as in the earlier chapters. The basis is the term algebra $\mathbb{P}(\mathrm{BCP}(A, \gamma)) = (\mathcal{C}(\mathrm{BCP}(A, \gamma)), +, \parallel, \mathbin{|\!\!\lfloor}, \mid, (a.\_)_{a \in A}, (\partial_H)_{H \subseteq A}, 0, 1)$.

The term deduction system for BCP(A, γ) is obtained by extending the term deduction system for $(\mathrm{BSP} + \mathrm{DH})(A)$ with deduction rules for the parallelism operators merge, left merge, communication merge, and is given in Table 7.4. The rules for the parallelism operators were already informally introduced in Section 7.2. The rules for left merge and communication merge each show a different part of the behavior of the merge operator.

TDS(BCP(A, γ))

TDS((BSP + DH)(A));

binary: $\_ \parallel \_$, $\_ \mathbin{|\!\!\lfloor} \_$, $\_ \mid \_$;

x, x', y, y';

$$\frac{x{\downarrow} \quad y{\downarrow}}{x \parallel y{\downarrow}} \qquad \frac{x{\downarrow} \quad y{\downarrow}}{x \mid y{\downarrow}}$$

$$\frac{x \overset{a}{\rightarrow} x'}{x \parallel y \overset{a}{\rightarrow} x' \parallel y} \qquad \frac{y \overset{a}{\rightarrow} y'}{x \parallel y \overset{a}{\rightarrow} x \parallel y'} \qquad \frac{x \overset{a}{\rightarrow} x'}{x \mathbin{|\!\!\lfloor} y \overset{a}{\rightarrow} x' \parallel y}$$

$$\frac{x \overset{a}{\rightarrow} x' \quad y \overset{b}{\rightarrow} y' \quad \gamma(a, b) = c}{x \parallel y \overset{c}{\rightarrow} x' \parallel y'} \qquad \frac{x \overset{a}{\rightarrow} x' \quad y \overset{b}{\rightarrow} y' \quad \gamma(a, b) = c}{x \mid y \overset{c}{\rightarrow} x' \parallel y'}$$

Table 7.4. Term deduction system for BCP(A, γ) (with $a, b, c \in A$).

Given the equational theory and the term deduction system, the development of the model and soundness, conservativity, and completeness results goes along the same lines as earlier. Figure 7.2 visualizes the conservativity results, introducing the Minimal Theory of Communicating Processes, MTCP (see Exercise 7.5.9).

Proposition 7.5.1 (Congruence) Bisimilarity is a congruence on term algebra $\mathbb{P}(\mathrm{BCP}(A, \gamma))$.

Proof The result follows immediately from the format of the deduction rules in Tables 4.2, 4.4, 6.10, and 7.4, and Theorem 3.2.7 (Congruence theorem). □

Definition 7.5.2 (Term model of BCP(A, γ)) The term model of BCP(A, γ) is the quotient algebra $\mathbb{P}(\text{BCP}(A, \gamma))_{/\leftrightarrows}$.

Theorem 7.5.3 (Soundness) Theory BCP(A, γ) is a sound axiomatization of the algebra $\mathbb{P}(\text{BCP}(A, \gamma))_{/\leftrightarrows}$, i.e., $\mathbb{P}(\text{BCP}(A, \gamma))_{/\leftrightarrows} \models \text{BCP}(A, \gamma)$.

Proof Exercise 7.5.6. □

Theorem 7.5.4 (Conservative ground-extension) Process theory BCP(A, γ) is a conservative ground-extension of process theory BSP(A).

Proof Exercise 7.5.7. □

Theorem 7.5.5 (Ground-completeness) The process theory BCP(A, γ) is a ground-complete axiomatization of the term model $\mathbb{P}(\text{BCP}(A, \gamma))_{/\leftrightarrows}$, i.e., for any two closed BCP(A, γ)-terms p and q, $\mathbb{P}(\text{BCP}(A, \gamma))_{/\leftrightarrows} \models p = q$ implies BCP$(A, \gamma) \vdash p = q$.

Proof Exercise 7.5.8. □

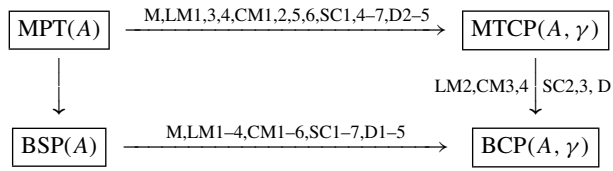

Fig. 7.2. Conservativity results for BCP(A, γ).

Exercises

7.5.1 Draw the transition system of the following BCP(A, γ)-terms, using the operational rules of Tables 4.2, 4.4, 6.10 and 7.4. Assume that $\gamma(a, b) = c$ is the only defined communication and that $H = \{a, b\}$.

(a) $a.a.1 \parallel b.b.1$;
(b) $\partial_H(a.a.1 \parallel b.b.1)$;

https://doi.org/10.1017/CBO9781139195003.011 Published online by Cambridge University Press

(c) $(a.1 + b.1) \parallel (a.1 + b.1)$;
(d) $\partial_H((a.1 + b.1) \parallel (a.1 + b.1))$;
(e) $a.a.a.1 \parallel a.a.a.1$.

7.5.2 Use the definition of bisimilarity and the operational rules of the term deduction system of $\mathrm{BCP}(A, \gamma)$ to show that $a.1 \between p \leftrightarrows a.p$, for any action $a \in A$ and closed $\mathrm{BCP}(A, \gamma)$-term p.

7.5.3 Let x, y, and z be arbitrary process terms. Give an example to show that equation $(x+y)\|z = x\|z+y\|z$ is not valid in $\mathbb{P}(\mathrm{BCP}(A, \emptyset))_{/\leftrightarrows}$. Note that the communication function is assumed to be empty, i.e., communication is not allowed.

7.5.4 Let x and y be process terms. Give an example to show that $\partial_H(x \parallel y) = \partial_H(x) \parallel \partial_H(y)$ is not valid in $\mathbb{P}(\mathrm{BCP}(A, \gamma))_{/\leftrightarrows}$. Is this equation valid if $\gamma = \emptyset$?

7.5.5 Define the n-fold parallel-composition operator $\_^{\parallel n}$ in the process theory $\mathrm{BCP}(A, \emptyset)$, with $n \in \mathbf{N}$, inductively as follows:

$$\begin{aligned} x^{\parallel 0} &= 1 \quad \text{and, for all } n \in \mathbf{N}, \\ x^{\parallel n+1} &= x^{\parallel n} \parallel x. \end{aligned}$$

Prove that for any action $a \in A$, $\mathrm{BCP}(A, \emptyset) \vdash (a.1)^{\parallel n} = a^n 1$, for all $n \in \mathbf{N}$, with a^n the n-fold action prefix of Notation 4.6.6. Show that $\mathrm{BCP}(A, \emptyset) \nvdash (a.b.1)^{\parallel n} = a^n b^n 1$.

7.5.6 Prove Theorem 7.5.3 (Soundness of $\mathrm{BCP}(A, \gamma)$).

7.5.7 Prove Theorem 7.5.4 (Conservative ground-extension).

7.5.8 Prove Theorem 7.5.5 (Ground-completeness of $\mathrm{BCP}(A, \gamma)$).

7.5.9 Develop the process theory $\mathrm{MTCP}(A, \gamma)$. That is, define the equational theory, prove an elimination and a conservativity result, and give a term model proving both soundness and ground-completeness. Also, prove that theory $\mathrm{BCP}(A, \gamma)$ is a conservative ground-extension of $\mathrm{MTCP}(A, \gamma)$.

7.6 Recursion, buffers, and bags

The fact that all occurrences of the parallel-composition operators can be eliminated from each closed $\mathrm{BCP}(A, \gamma)$-term does not imply that the addition of parallel composition is without consequences in a context with recursion. In fact, using parallel composition (even without communication), it is possible to give a finite guarded recursive specification over the signature of the theory $\mathrm{BCP}(A, \gamma)$ of a process that is not finitely definable over any of the earlier theories.

To extend $\mathrm{BCP}(A, \gamma)$ with recursion, the theory is extended with constants

https://doi.org/10.1017/CBO9781139195003.011 Published online by Cambridge University Press

$\mu X.E$ for any recursion variable X in a recursive specification E, and with axioms corresponding to the equations in all the recursive specifications of interest. The resulting theory is theory $\mathrm{BCP}_{\mathrm{rec}}(A,\gamma)$. The term deduction system $TDS(\mathrm{BCP}_{\mathrm{rec}}(A,\gamma))$ is obtained by merging the term deduction systems $TDS(\mathrm{BSP}_{\mathrm{rec}}(A))$ of Table 5.2 and $TDS(\mathrm{BCP}(A,\gamma))$ of Table 7.4. The resulting term model is $\mathbb{P}(\mathrm{BCP}_{\mathrm{rec}}(A,\gamma))_{/\leftrightarrows}$. There are no elimination and ground-completeness results, but the extension with recursion is conservative.

Recursion principles are needed to allow for meaningful equational reasoning in a context with recursion. The development goes along the lines of Section 6.6, which discusses the extension of $\mathrm{TSP}(A)$ with recursion. The extension of $\mathrm{BCP}(A,\gamma)$ and $\mathrm{BCP}_{\mathrm{rec}}(A,\gamma)$ with projection, needed for AIP and AIP^-, is straightforward. Definition 5.5.8 (Guardedness) carries over to the current context. The reasoning leading to Theorem 7.6.2 (Recursion principles) uses the following generalization of Proposition 5.5.26 (HNF property), which shows that every guarded $(\mathrm{BCP}+\mathrm{PR})_{\mathrm{rec}}(A,\gamma)$-term can be rewritten into a head normal form (see Definition 5.5.24).

Proposition 7.6.1 (HNF property) Process theory $(\mathrm{BCP}+\mathrm{PR})_{\mathrm{rec}}(A,\gamma)$ satisfies the HNF property.

Proof It needs to be shown that every guarded $(\mathrm{BCP}+\mathrm{PR})_{\mathrm{rec}}(A,\gamma)$-term can be rewritten into a head normal form. It may be assumed that this term, say s, is completely guarded. The proof is by induction on the structure of s. Compared to Proposition 5.5.26, there are four extra cases to consider:

- $s \equiv \partial_H(s')$, for some completely guarded $(\mathrm{BCP}+\mathrm{PR})_{\mathrm{rec}}(A,\gamma)$-term s' and subset of actions $H \subseteq A$;
- $s \equiv s' \between s''$, for completely guarded $(\mathrm{BCP}+\mathrm{PR})_{\mathrm{rec}}(A,\gamma)$-terms s' and s'';
- $s \equiv s'|s''$, for completely guarded $(\mathrm{BCP}+\mathrm{PR})_{\mathrm{rec}}(A,\gamma)$-terms s' and s'';
- $s \equiv s' \parallel s''$, for completely guarded $(\mathrm{BCP}+\mathrm{PR})_{\mathrm{rec}}(A,\gamma)$-terms s' and s''.

These cases are not difficult. See Exercise 7.6.1. □

Theorem 7.6.2 (Recursion principles) The principles RDP, RDP^-, RSP, and AIP^- are valid in the term model $\mathbb{P}((\mathrm{BCP}+\mathrm{PR})_{\mathrm{rec}}(A,\gamma))_{/\leftrightarrows}$. Principle AIP is not valid in this model. RDP, RDP^-, and RSP are also valid in model $\mathbb{P}(\mathrm{BCP}_{\mathrm{rec}}(A,\gamma))_{/\leftrightarrows}$.

https://doi.org/10.1017/CBO9781139195003.011 Published online by Cambridge University Press

A few results derived earlier deserve attention in a context with recursion. First, recall Proposition 7.4.2 (Communication with the empty process). It shows under what conditions a communication with the empty process results in successful or unsuccessful termination. If a process has a 1 summand, then it can be derived that the communication of that process with the empty process results in the empty process, i.e., in successful termination. It cannot in general be derived that the absence of a 1 summand in the process results in inaction. However, the second part of Proposition 7.4.2, concerning the absence of a 1 summand, holds for guarded terms.

Proposition 7.6.3 (Communication with the empty process) Assume that s is a guarded $\mathrm{BCP}_{\mathrm{rec}}(A, \gamma)$-term. If term s does not contain a 1 summand, then $\mathrm{BCP}_{\mathrm{rec}}(A, \gamma) \vdash 1 \mid s = 0$.

Proof Based on Proposition 7.6.1 (HNF property), let t be a head normal form such that $\mathrm{BCP}_{\mathrm{rec}}(A, \gamma) \vdash s = t$. It follows from Proposition 5.5.25 (Head normal forms) and the fact that s does not have a 1 summand, that there is a natural number n, and that there are, for any $i < n$, $a_i \in A$ and $\mathrm{BCP}(A, \gamma)$-terms t_i such that $\mathrm{BCP}_{\mathrm{rec}}(A, \gamma) \vdash t = \sum_{i<n} a_i.t_i$. It is now possible to prove that s satisfies the condition in the second property in Proposition 7.4.2. However, it also follows directly that $\mathrm{BCP}_{\mathrm{rec}}(A, \gamma) \vdash 1 \mid s = 1 \mid t = 1 \mid \sum_{i<n} a_i.t_i = \sum_{i<n} 1 \mid a_i.t_i = \sum_{i<n} 0 = 0$. □

Second, recall Proposition 7.4.8 (Free-Merge Axiom), stating that the Free-Merge Axiom can be derived from the axioms of theory $\mathrm{BCP}(A, \emptyset)$ for all closed $\mathrm{BCP}(A, \emptyset)$-terms. This proposition can be generalized to all guarded $\mathrm{BCP}_{\mathrm{rec}}(A, \emptyset)$-terms in the current context with recursion. Also the Handshaking Axiom (see Table 7.3) is derivable for all guarded $\mathrm{BCP}_{\mathrm{rec}}(A, \gamma)$-terms when γ is a handshaking communication function.

As an example of calculations with recursive specifications in $\mathrm{BCP}_{\mathrm{rec}}(A, \gamma)$, let us consider *buffers* of finite capacity. Recall Definition 7.3.1 (Standard communication). Assume that the set of data elements D is the set of bits $\{0, 1\}$. In the minimal theory $\mathrm{MPT}(A)$ with recursion, a *one-bit buffer* with input port i and output port o can be specified as follows:

$$Buf1 = 1 + i?0.o!0.Buf1 + i?1.o!1.Buf1.$$

Note that in this specification, an empty buffer has an option to terminate. The reason to include this option is that an environment using the buffer can decide to terminate. Absence of this initial termination option would prevent such termination; see Exercise 7.6.7.

In general, a one-place buffer over a finite data set D is given as follows:

https://doi.org/10.1017/CBO9781139195003.011 Published online by Cambridge University Press

$$Buf1 = 1 + \sum_{d \in D} i?d.o!d.Buf1.$$

To describe a buffer with capacity two, a specification with two equations is needed (one parameterized with a data element, meaning that the actual number of equations in the recursive specification equals the cardinality of D plus one):

$$\begin{aligned} Buf2 &= 1 + \sum_{d \in D} i?d.B_d \quad \text{and, for all } d \in D, \\ B_d &= o!d.Buf2 + \sum_{e \in D} i?e.o!d.B_e. \end{aligned}$$

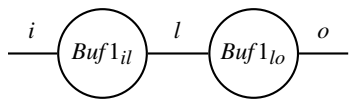

Fig. 7.3. A sequence of two one-place buffers.

Consider now a communication network consisting of two buffers of capacity one, $Buf1_{il}$ with input port i and output port l (short for *link*), and $Buf1_{lo}$ with input port l and output port o; see Figure 7.3. The two components are specified as above:

$$\begin{aligned} Buf1_{il} &= 1 + \sum_{d \in D} i?d.l!d.Buf1_{il} \quad \text{and} \\ Buf1_{lo} &= 1 + \sum_{d \in D} l?d.o!d.Buf1_{lo}. \end{aligned}$$

It is interesting to consider the parallel composition of the two processes $Buf1_{il}$ and $Buf1_{lo}$. Assume communication is specified by the standard communication function γ_S of Definition 7.3.1. To enforce communication in the parallel composition, encapsulation of halves of internal communication actions is required. Thus, put $H = \{l?d, l!d \mid d \in D\}$ and consider the process $\partial_H(Buf1_{il} \parallel Buf1_{lo})$. This process can be interpreted as a two-place buffer, with an internal communication port l. Values from D are passed along this port, using the synchronous communication mechanism of the theory. A guarded recursive specification can be derived for this process. Let $X = \partial_H(Buf1_{il} \parallel Buf1_{lo})$ and, for each $d \in D$, let $X_d = \partial_H(Buf1_{il} \parallel o!d.Buf1_{lo})$.

$$\begin{aligned} &\mathrm{BCP}_{\mathrm{rec}}(A, \gamma_S) \vdash \\ &\quad X = \partial_H\left(Buf1_{il} \mathbin{\lfloor\!\lfloor} Buf1_{lo}\right) + \partial_H\left(Buf1_{lo} \mathbin{\lfloor\!\lfloor} Buf1_{il}\right) \\ &\qquad + \partial_H(Buf1_{il} \mid Buf1_{lo}) \\ &\qquad = \partial_H\left(\sum_{d \in D} i?d.(l!d.Buf1_{il} \parallel Buf1_{lo})\right) + 0 + 1 \end{aligned}$$

https://doi.org/10.1017/CBO9781139195003.011 Published online by Cambridge University Press

$$
\begin{aligned}
&= 1 + \sum_{d\in D} i?d.\partial_H(l!d.Buf1_{il} \between Buf1_{lo} \\
&\qquad\qquad + Buf1_{lo} \between l!d.Buf1_{il} \\
&\qquad\qquad + l!d.Buf1_{il} \mid Buf1_{lo} \\
&\qquad\qquad)\\
&= 1 + \sum_{d\in D} i?d.(0 + 0 + l?d.\partial_H(Buf1_{il} \parallel o!d.Buf1_{lo}))\\
&= 1 + \sum_{d\in D} i?d.l?d.X_d
\end{aligned}
$$

and

$$
\begin{aligned}
&\mathrm{BCP}_{\mathrm{rec}}(A,\gamma_S) \vdash X_d\\
&\quad = \partial_H\left(Buf1_{il} \between o!d.Buf1_{lo}\right) + \partial_H\left(o!d.Buf1_{lo} \between Buf1_{il}\right)\\
&\qquad + \partial_H\left(Buf1_{il} \mid o!d.Buf1_{lo}\right)\\
&\quad = \sum_{e\in D} i?e.\partial_H\left(l!e.Buf1_{il} \parallel o!d.Buf1_{lo}\right)\\
&\qquad + o!d.\partial_H\left(Buf1_{il} \parallel Buf1_{lo}\right) + 0\\
&\quad = \sum_{e\in D} i?e.(0 + o!d.\partial_H\left(l!e.Buf1_{il} \parallel Buf1_{lo}\right) + 0) + o!d.X\\
&\quad = \sum_{e\in D} i?e.o!d.(0 + 0 + l?e.\partial_H\left(Buf1_{il} \parallel o!e.Buf1_{lo}\right)) + o!d.X\\
&\quad = \sum_{e\in D} i?e.o!d.l?e.X_e + o!d.X.
\end{aligned}
$$

Thus, a system of two connected one-place buffers is given by recursive specification:

$$
\begin{aligned}
X &= 1 + \sum_{d\in D} i?d.l?d.X_d \quad \text{and, for all } d \in D,\\
X_d &= o!d.X + \sum_{e\in D} i?e.o!d.l?e.X_e.
\end{aligned}
$$

Recall from Section 6.7 the concept of skip operators. It can be seen that skipping the internal communication actions in this specification gives the two-place buffer between ports i and o, i.e., with $I = \{l?d \mid d \in D\}$,

$$
\begin{aligned}
&((\mathrm{BCP} + \mathrm{EI})_{\mathrm{rec}} + \mathrm{RSP})(A,\gamma_S) \vdash\\
&\qquad \varepsilon_I(\partial_H(Buf1_{il} \parallel Buf1_{lo})) = Buf2.
\end{aligned}
$$

The next chapter introduces a notion of abstraction, which can be used to obtain a similar result (in a conceptually more elegant way); see Exercise 8.8.1.

This example once more shows the general form of a system describing a communication network: a number of component processes, in the scope of an encapsulation operator that blocks isolated send and receive actions on internal ports. This allows only communication actions on these internal ports and blocks interaction with the environment over these ports.

The following considers an example of a recursive specification of a process in $\mathrm{BCP}_{\mathrm{rec}}(A,\emptyset)$, i.e., the theory with a free merge where communication is not possible. Notwithstanding, standard notation for inputs and outputs (send and receive actions) established earlier is used. The process *bag* of unbounded

https://doi.org/10.1017/CBO9781139195003.011 Published online by Cambridge University Press

capacity is able to input arbitrary elements of a finite data set D at port i and output elements that have been input previously in any order, at port o. Thus, input order is not important, but the number of specific elements present has to be kept track of. To keep things simple, first consider only bits, so $D = \{0, 1\}$. This means there are input actions $i?0$ and $i?1$, and output actions $o!0$ and $o!1$.

An (infinite) recursive specification over the signature of the minimal theory MPT(A) has variables $B_{n,m}$ that denote the state of the bag with n zeroes and m ones. The following equations exist for all $n, m \geq 0$, and together form the (linear) recursive specification E.

$$\begin{aligned}
B_{0,0} &= 1 + i?0.B_{1,0} + i?1.B_{0,1},\\
B_{0,m+1} &= o!1.B_{0,m} + i?0.B_{1,m+1} + i?1.B_{0,m+2},\\
B_{n+1,0} &= o!0.B_{n,0} + i?0.B_{n+2,0} + i?1.B_{n+1,1},\\
B_{n+1,m+1} &= o!0.B_{n,m+1} + o!1.B_{n+1,m} +\\
&\qquad + i?0.B_{n+2,m+1} + i?1.B_{n+1,m+2}.
\end{aligned}$$

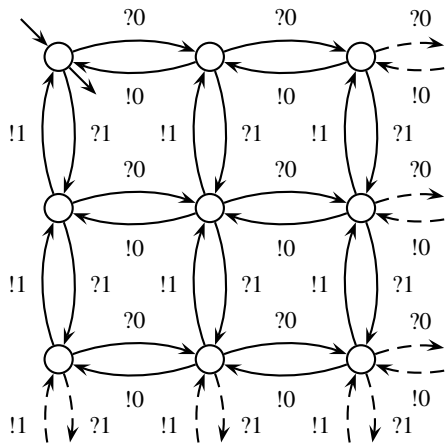

Fig. 7.4. The transition system of a bag of bits.

Figure 7.4 visualizes the transition system of the above specification of the bag of bits, using obvious abbreviations for input and output actions.

The following gives a *finite* recursive specification F of a bag, in BCP($A, \emptyset$) with recursion. This specification has just one variable, Bag.

$$Bag = 1 + i?0.(Bag \parallel o!0.1) + i?1.(Bag \parallel o!1.1).$$

It can be seen immediately that this is a guarded recursive specification.

Proposition 7.6.4 (Bags)

$$(\mathrm{BCP} + \mathrm{FMA} + E + F + \mathrm{RSP})(A, \emptyset) \vdash Bag = B_{0,0}.$$

https://doi.org/10.1017/CBO9781139195003.011 Published online by Cambridge University Press

Proof Define the processes $D_{n,m}$, for $n, m \geq 0$, as follows:

$$D_{n,m} = Bag \parallel (o!0)^n 1 \parallel (o!1)^m 1,$$

where as before a^n for action a is the n-fold action prefix of Notation 4.6.6. The first steps in the first derivation below show that $(\mathrm{BCP}+\mathrm{FMA}+F)(A,\emptyset)\vdash D_{0,0} = Bag$. Therefore, to prove the theorem, it is sufficient to show that the processes $D_{n,m}$ form a solution of E. Consider the equations one by one. First,

$$\begin{aligned}
&(\mathrm{BCP}+\mathrm{FMA}+F)(A,\emptyset)\vdash\\
&\quad D_{0,0} = Bag \parallel (o!0)^0 1 \parallel (o!1)^0 1\\
&\qquad = Bag \parallel 1 \parallel 1\\
&\qquad = Bag\\
&\qquad = 1 + i?0.(Bag \parallel o!0.1) + i?1.(Bag \parallel o!1.1)\\
&\qquad = 1 + i?0.(Bag \parallel (o!0)^1 1 \parallel 1) + i?1.(Bag \parallel 1 \parallel (o!1)^1 1)\\
&\qquad = 1 + i?0.D_{1,0} + i?1.D_{0,1}.
\end{aligned}$$

The second derivation uses Axiom M. Since communication is not possible and one of the terms does not contain a 1 summand, the communication-merge term is skipped. This derivation also uses laws of standard concurrency and the fact that, for each atomic action $a \in A$ and natural number $k \in \mathbf{N}$, the equality $a^k 1 \parallel a.1 = a^{k+1}1$ is derivable (see Exercise 7.4.4).

$$\begin{aligned}
&(\mathrm{BCP}+\mathrm{FMA}+F)(A,\emptyset)\vdash\\
&\quad D_{0,m+1} = Bag \parallel 1 \parallel (o!1)^{m+1} 1\\
&\qquad = Bag \parallel (o!1)^{m+1} 1\\
&\qquad = Bag \mathbin{\lfloor\!\lfloor} (o!1)^{m+1} 1 + (o!1)^{m+1} 1 \mathbin{\lfloor\!\lfloor} Bag\\
&\qquad = (1 + i?0.D_{1,0} + i?1.D_{0,1}) \mathbin{\lfloor\!\lfloor} (o!1)^{m+1} 1\\
&\qquad\qquad + (o!1)^{m+1} 1 \mathbin{\lfloor\!\lfloor} Bag\\
&\qquad = i?0.(D_{1,0} \parallel (o!1)^{m+1} 1) + i?1.(D_{0,1} \parallel (o!1)^{m+1} 1)\\
&\qquad\qquad + o!1.((o!1)^m 1 \parallel Bag)\\
&\qquad = i?0.D_{1,m+1} + i?1.D_{0,m+2} + o!1.D_{0,m}.
\end{aligned}$$

The third derivation, for $D_{n+1,0}$, is left as an exercise to the reader, because it is similar to the previous one.

Finally, the last derivation starts with an application of the expansion theorem for the free merge (Theorem 7.4.9). Again, the communication-merge term is skipped, as no termination is possible.

$$\begin{aligned}
&(\mathrm{BCP}+\mathrm{FMA}+F)(A,\emptyset) \vdash D_{n+1,m+1}\\
&\quad = Bag \parallel (o!0)^{n+1} 1 \parallel (o!1)^{m+1} 1\\
&\quad = Bag \mathbin{\lfloor\!\lfloor} ((o!0)^{n+1} 1 \parallel (o!1)^{m+1} 1)\\
&\qquad + (o!0)^{n+1} 1 \mathbin{\lfloor\!\lfloor} (Bag \parallel (o!1)^{m+1} 1)\\
&\qquad + (o!1)^{m+1} 1 \mathbin{\lfloor\!\lfloor} (Bag \parallel (o!0)^{n+1} 1)
\end{aligned}$$

https://doi.org/10.1017/CBO9781139195003.011 Published online by Cambridge University Press

$$
\begin{aligned}
&= (1 + i?0.D_{1,0} + i?1.D_{0,1}) \between ((o!0)^{n+1}1 \parallel (o!1)^{m+1}1) \\
&\qquad + (o!0)^{n+1}1 \between (Bag \parallel (o!1)^{m+1}1) \\
&\qquad + (o!1)^{m+1}1 \between (Bag \parallel (o!0)^{n+1}1) \\
&= i?0.(D_{1,0} \parallel (o!0)^{n+1}1 \parallel (o!1)^{m+1}1) \\
&\qquad + i?1.(D_{0,1} \parallel (o!0)^{n+1}1 \parallel (o!1)^{m+1}1) \\
&\qquad + o!0.((o!0)^{n}1 \parallel Bag \parallel (o!1)^{m+1}1) \\
&\qquad + o!1.((o!1)^{m}1 \parallel Bag \parallel (o!0)^{n+1}1) \\
&= i?0.D_{n+2,m+1} + i?1.D_{n+1,m+2} + o!0.D_{n,m+1} + o!1.D_{n+1,m}.
\end{aligned}
$$

The above derivations show that processes $D_{n,m}$ form a solution of specification E. Hence, by RSP, it follows that $(\mathrm{BCP} + \mathrm{FMA} + E + F + \mathrm{RSP})(A, \emptyset) \vdash D_{n,m} = B_{n,m}$, and in particular that $(\mathrm{BCP} + \mathrm{FMA} + E + F + \mathrm{RSP})(A, \emptyset) \vdash Bag = B_{0,0}$. □

The above proposition shows that there is a specification for a bag over a data set of two elements in the theory $\mathrm{BCP}_{\mathrm{rec}}(A, \emptyset)$ using just one guarded recursive equation. Such a finite guarded recursive specification does not exist in the theory $\mathrm{TSP}_{\mathrm{rec}}(A)$. For a proof of this fact, the interested reader is referred to (Bergstra & Klop, 1984b). Recall Definition 5.7.5 (Definability). An interesting consequence of these observations is that the parallel composition does add expressive power, even in the absence of communication. It is possible to finitely define a process that cannot be finitely defined when parallel composition is absent, not even when sequential composition is present. More precisely, when A contains at least four actions, there are processes that are finitely definable over $\mathrm{BCP}(A, \emptyset)$ that are not finitely definable over $\mathrm{TSP}(A)$; the requirement on A is necessary because the mentioned proof in (Bergstra & Klop, 1984b) essentially uses the fact that the specification of the bag with two data elements uses four actions.

The expressive power of finite guarded recursive specifications over the signatures of theories $\mathrm{BCP}(A, \gamma)$ and $\mathrm{TSP}(A)$ is probably incomparable. To show this, it remains to give a finite guarded recursive specification over the signature of $\mathrm{TSP}(A)$ that defines a process that cannot be finitely defined over $\mathrm{BCP}(A, \gamma)$; see Exercise 7.6.10. Note that the theories with general recursion, $\mathrm{BCP}_{\mathrm{rec}}(A, \gamma)$ and $\mathrm{TSP}_{\mathrm{rec}}(A)$, are equally expressive (as defined in Definition 5.7.2 (Expressiveness)). Already in the basic theory $\mathrm{BSP}_{\mathrm{rec}}(A)$, it is possible to specify all countable computable processes; this does not change with the extensions with sequential composition or parallel composition. Without recursion, $\mathrm{BCP}(A, \gamma)$ and $\mathrm{TSP}(A)$ are also equally expressive. Both theories extend the basic theory $\mathrm{BSP}(A)$ and all closed $\mathrm{BCP}(A, \gamma)$- and $\mathrm{TSP}(A)$-terms can be rewritten into equivalent $\mathrm{BSP}(A)$-terms, which makes these three theories equally expressive.

In case the data set D is a finite set, not necessarily containing only two elements, the generalized-choice notation can be used to obtain an even more compact equation for the specification of a bag over D:

$$Bag = 1 + \sum_{d \in D} i?d.(Bag \parallel o!d.1).$$

The equation for a bag can be specialized to the case where the data type contains exactly one element, say d. The bag in this case is just a counter, which can be specified as follows when assuming that $i?d \equiv plus$ and $o!d \equiv minus$:

$$Counter2 = 1 + plus.(Counter2 \parallel minus.1).$$

Recall that Exercise 6.6.5 also introduced a specification of a counter, using sequential composition. In order to reason about the relation between these two counter specifications, a theory is needed where both sequential composition and parallel composition are present. This theory, TCP, the Theory of Communicating Processes, is treated in the following section.

Exercises

7.6.1 Prove Proposition 7.6.1 (HNF property).

7.6.2 Let $n \geq 1$. Give a specification of an n-place buffer $Buf n$.

7.6.3 Describe a biscuit-tin (with unbounded capacity) with two kinds of biscuits by means of a guarded recursive specification over theory $\mathrm{BCP}(A, \emptyset)$ with recursion.

7.6.4 Consider the extension of $\mathrm{BCP}(A, \gamma)$ with projection. For the process Bag defined above (for an arbitrary finite data set D), calculate $\pi_1(Bag)$, $\pi_2(Bag)$, and $\pi_3(Bag)$.

7.6.5 Sketch the transition system of a bag over data set $D = \{0, 1, 2\}$.

7.6.6 Verify that

$$((\mathrm{BCP} + \mathrm{EI})_{\mathrm{rec}} + \mathrm{RSP})(A, \gamma_S) \vdash \varepsilon_I(\partial_H(Buf1_{il} \parallel Buf1_{lo})) = Buf2,$$

using among others the axioms of Table 6.9.

7.6.7 Consider two users that communicate via the one-place buffer $Buf1$ specified in this section. One user sends two data items 0 over port i and one receives these two data items over port o. Both users terminate successfully after sending resp. receiving their data. Specify this system, and prove that it can terminate successfully. Show that it cannot terminate successfully if the termination option in the defining

https://doi.org/10.1017/CBO9781139195003.011 Published online by Cambridge University Press

equation for $Buf1$ is omitted. Argue that the system even with the initial termination option of the buffer cannot terminate when the buffer is not empty.

7.6.8 Consider again the communication network of Figure 7.3, but now with the processes S and R instead of $Buf1_{il}$ and $Buf1_{lo}$, given by the equations:

$$S = 1 + \sum_{d \in D} i?d.l!d.l?ack.S,$$
$$R = 1 + \sum_{d \in D} l?d.o!d.l!ack.R.$$

In these equations, $ack \notin D$ is a special element denoting an *acknowledgement*. Let $H = \{l!d, l?d \mid d \in D \cup \{ack\}\}$. Find a recursive equation for process $\partial_H(S \parallel R)$.

7.6.9 Let p be a closed BCP$(A, \emptyset)$-term. The *replication* of p, in the literature on process algebra often denoted $!p$, is the process term $p \parallel p \parallel p \parallel \ldots$. This process cannot be defined (in the sense of Definition 5.7.5 (Definability)) by equation $X = p \parallel X$ with X some recursion variable, as this equation is unguarded, and has infinitely many solutions in the term model of theory BCP$_{\mathrm{rec}}(A, \emptyset)$. However, the equation $X = p \between X + p \mid 1$ can be used for the intended purpose (in the theory without communication). Argue that this equation is guarded, and yields the replication of p.

7.6.10 Recall Definition 5.7.5 (Definability). Give a finite guarded recursive specification over the signature of theory TSP(A) that defines a process that cannot be finitely defined over theory BCP(A, γ), or, alternatively, prove that TSP(A) allows to finitely define strictly fewer processes than BCP(A, γ). Inform the authors about your answer.

7.7 The process theory TCP and further extensions

It is often useful to have both sequential composition and parallel composition present in the same theory. The process theory TCP, the Theory of Communicating Processes, is the union of the process theories BCP and TSP; see Table 7.5. The axioms of TCP(A, γ) are the axioms of BCP(A, γ) and TSP(A), see Tables 7.1 and 6.1, with one additional Axiom of Standard Concurrency.

It is straightforward to obtain elimination and conservativity results with respect to theory BSP(A). Figure 7.5 visualizes the conservativity results for the basic theories with parallel composition. As before, the extra Axiom of Standard Concurrency is derivable for closed terms.

$$\frac{\text{TCP}(A,\gamma)}{\text{TSP}(A), \text{BCP}(A,\gamma);}$$
$$\frac{-}{x;}$$
$$x \between 0 = x \cdot 0 \quad \text{SC8}$$

Table 7.5. The process theory $\text{TCP}(A, \gamma)$.

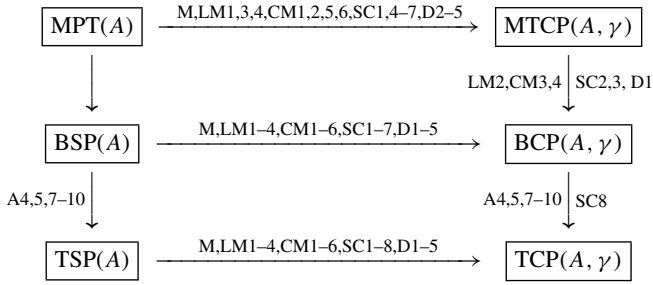

Fig. 7.5. Conservativity results for $\text{TCP}(A, \gamma)$.

Theorem 7.7.1 (Standard concurrency) For closed $\text{TCP}(A, \gamma)$-terms, Axiom SC8 is derivable from the other axioms of $\text{TCP}(A, \gamma)$.

Proof By structural induction. See Exercise 7.7.2. □

The term deduction system underlying the term model of theory $\text{TCP}(A, \gamma)$ is just the union of the two term deduction systems for $\text{TSP}(A)$ and $\text{BCP}(A, \gamma)$. It is a useful exercise to show that the resulting term model admits a soundness and a ground-completeness theorem.

Extension of $\text{TCP}(A, \gamma)$ with other features does not present difficulties. For instance, it is straightforward to extend the theory with projection operators π_n for all $n \in \mathbf{N}$, resulting in the theory $(\text{TCP} + \text{PR})(A, \gamma)$, or with renaming (see Section 6.7), resulting in $(\text{TCP} + \text{RN})(A, \gamma)$.

Example 7.7.2 (CSP parallel composition and communication) As an example of a definition in the theory $(\text{TCP} + \text{RN})(A, \gamma)$, consider the parallel composition operator of CSP, see (Hoare, 1985). For a given set of action names S, the actions in S must synchronize as much as possible, where different actions with the same name may synchronize, resulting in one occurrence of the same action. The difficulty is to ensure that these actions do not occur by themselves when they should synchronize. In order to achieve this,

assume that the set of actions A is divided into two parts: a set of names N and a set of communications N_c such that for each $n \in N$ there is exactly one $n_c \in N_c$. Now take the communication function γ that has $\gamma(n, n) = n_c$ and is not defined otherwise, and the renaming function f with $f(n_c) = n$. Then, CSP-style parallel composition $\|_S^{\mathrm{CSP}}$, with $S \subseteq N$, can be defined by axiom

$$x \|_S^{\mathrm{CSP}} y = \rho_f(\partial_{S \cup (N_c \setminus S_c)}(x \| y)),$$

where $S_c = \{n_c \mid n \in S\}$. As an example, if $S = \{a\}$ and $a, b, c \in N$, then it can be derived that

$$b.a.b.a.1 \|_S^{\mathrm{CSP}} a.c.1 = b.a.(b.c.0 + c.b.0).$$

Also the extension with recursion follows the standard steps. To illustrate reasoning in $\mathrm{TCP}(A, \gamma)$, consider the counter specifications given in the previous section and in Exercise 6.6.5. Since there is no communication and the counter specifications use recursion, and because the equational reasoning uses projection operators, the precise context is the theory $(\mathrm{TCP} + \mathrm{PR})_{\mathrm{rec}}(A, \emptyset)$. In the previous section, the following recursive equation for a counter was given:

$$\mathit{Counter}2 = 1 + \mathit{plus}.(\mathit{Counter}2 \| \mathit{minus}.1).$$

In Exercise 6.6.5, a different specification was given, namely,

$$\begin{aligned} \mathit{Counter} &= 1 + T \cdot \mathit{Counter}, \\ T &= \mathit{plus}.T', \\ T' &= \mathit{minus}.1 + T \cdot T'. \end{aligned}$$

It can be shown that these two specifications are equal. Recall that also Exercise 5.6.3 gives a(n infinite) recursive specification of a counter. One way to prove the equality of the above two specifications is by showing that the two processes both satisfy this infinite specification and to apply RSP, which is essentially the technique applied most often so far. To illustrate a different proof technique, the desired equality can also be proven via AIP^-.

Proposition 7.7.3 (Counters) $((\mathrm{TCP} + \mathrm{PR})_{\mathrm{rec}} + \mathrm{AIP}^-)(A, \emptyset) \vdash \mathit{Counter} = \mathit{Counter}2$.

Proof The proof shows that

$$\begin{aligned} &((\mathrm{TCP} + \mathrm{PR})_{\mathrm{rec}} + \mathrm{AIP}^-)(A, \emptyset) \vdash \\ &\qquad (T')^k \cdot \mathit{Counter} = \mathit{Counter}2 \| \mathit{minus}^k 1, \end{aligned}$$

for every natural number $k \in \mathbf{N}$, where x^k for any term x is the k-fold sequential composition of Exercise 6.2.3. Note that this equation reduces to

$$((\mathrm{TCP} + \mathrm{PR})_{\mathrm{rec}} + \mathrm{AIP}^-)(A, \emptyset) \vdash \mathit{Counter} = \mathit{Counter}2,$$

https://doi.org/10.1017/CBO9781139195003.011 Published online by Cambridge University Press

if k equals zero. By AIP$^-$, it is sufficient to show that all finite projections of the processes in the desired equations are derivably equal. This can be proven by induction. Observe that the recursive specification of *Counter* can be rewritten as follows:

$$\begin{aligned}&(\mathrm{TCP}+\mathrm{PR})_{\mathrm{rec}}(A,\emptyset)\vdash\\&\quad Counter = 1 + plus.T'\cdot Counter,\\&\quad T' \quad\;\; = minus.1 + plus.(T')^2.\end{aligned}$$

The basis of the induction considers projections of depth 0. First, assume $k = 0$. Then,

$$\begin{aligned}&(\mathrm{TCP}+\mathrm{PR})_{\mathrm{rec}}(A,\emptyset)\vdash\\&\quad \pi_0(Counter) = \pi_0(1 + plus.T'\cdot Counter) = 1\end{aligned}$$

and

$$\begin{aligned}&(\mathrm{TCP}+\mathrm{PR})_{\mathrm{rec}}(A,\emptyset)\vdash\\&\quad \pi_0(Counter2) = \pi_0(1 + plus.(Counter2 \parallel minus.1)) = 1.\end{aligned}$$

Similarly, if $k > 0$, then

$$\begin{aligned}&(\mathrm{TCP}+\mathrm{PR})_{\mathrm{rec}}(A,\emptyset)\vdash\\&\quad \pi_0((T')^k\cdot Counter) = 0 = \pi_0(Counter2 \parallel minus^k 1).\end{aligned}$$

For the inductive step, assume that the desired result holds for some $n \in \mathbf{N}$. Then, the following derivation can be made for $n + 1$ if k equals 0:

$$\begin{aligned}&(\mathrm{TCP}+\mathrm{PR})_{\mathrm{rec}}(A,\emptyset)\vdash\\&\quad \pi_{n+1}(Counter) = \pi_{n+1}(1 + plus.T'\cdot Counter)\\&\qquad\qquad = 1 + plus.\pi_n(T'\cdot Counter)\\&\qquad\qquad = 1 + plus.\pi_n(Counter2 \parallel minus.1)\\&\qquad\qquad = \pi_{n+1}(1 + plus.(Counter2 \parallel minus.1))\\&\qquad\qquad = \pi_{n+1}(Counter2).\end{aligned}$$

The final part of the proof uses the equality proven in Exercise 7.4.4. If $k = l + 1 > 0$ for some natural number $l \in \mathbf{N}$,

$$\begin{aligned}&(\mathrm{TCP}+\mathrm{PR})_{\mathrm{rec}}(A,\emptyset)\vdash \pi_{n+1}((T')^{l+1}\cdot Counter)\\&\quad = \pi_{n+1}(minus.(T')^l\cdot Counter + plus.(T')^{l+2}\cdot Counter)\\&\quad = minus.\pi_n((T')^l\cdot Counter) + plus.\pi_n((T')^{l+2}\cdot Counter\\&\quad = minus.\pi_n(Counter2 \parallel minus^l 1)\\&\qquad\qquad + plus.\pi_n(Counter2 \parallel minus^{l+2}1)\\&\quad = \pi_{n+1}(minus.(Counter2 \parallel minus^l 1)\\&\qquad\qquad + plus.(Counter2 \parallel minus^{l+2}1))\\&\quad = \pi_{n+1}(minus^{l+1}1 \between Counter2\\&\qquad\qquad + plus.((Counter2 \parallel minus.1) \parallel minus^{l+1}1))\end{aligned}$$

https://doi.org/10.1017/CBO9781139195003.011 Published online by Cambridge University Press

$$
\begin{aligned}
&= \pi_{n+1}(minus^{l+1}1 \between Counter2 \\
&\qquad + plus.(Counter2 \parallel minus.1) \between minus^{l+1}1) \\
&= \pi_{n+1}(minus^{l+1}1 \between Counter2 + Counter2 \between minus^{l+1}1) \\
&= \pi_{n+1}(Counter2 \parallel minus^{l+1}1).
\end{aligned}
$$

□

At this point, it is interesting to consider some expressiveness aspects. The results of (Bergstra & Klop, 1984b) imply that there are processes that can be defined by a finite guarded recursive specification over the signature of theory TCP(A, γ), for a certain γ, but that cannot be defined by a finite guarded recursive specification over the signature of TCP$(A, \emptyset)$. Also an unbounded FIFO (First-In-First-Out) queue is an example of a process that is finitely definable over TCP(A, γ) when general communication is allowed but not over TCP$(A, \emptyset)$. The queue example is discussed in more detail below; in particular, Exercise 7.7.10 gives a finite guarded recursive specification over the signature of theory TCP(A, γ). These observations imply that finite guarded recursive specifications over the signature of theory TCP(A, γ) for a general communication function have more expressive power than those over the signature of TCP$(A, \emptyset)$. This illustrates the added value of communication when compared to parallel composition without communication. As before, when general recursion is allowed, the two variants $\mathrm{TCP}_{\mathrm{rec}}(A, \gamma)$ and $\mathrm{TCP}_{\mathrm{rec}}(A, \emptyset)$ with and without communication are equally expressive in the sense of Definition 5.7.2 (Expressiveness); they both allow the specification of all countable computable processes. Also TCP(A, γ) and TCP$(A, \emptyset)$, i.e., the theories without recursion, are equally expressive in the sense of this definition, because, independent of the definition of γ, the set of closed TCP(A, γ)-terms contains all closed BSP(A)-terms, and, again independent of the definition of γ, all closed TCP(A, γ)-terms can always be rewritten into a closed BSP(A)-term. Hence, both the generic theory TCP(A, γ) and the variant with the free merge are equally expressive as BSP(A) (and therefore equally expressive as theories such as TSP(A) and BCP(A, γ)).

As already suggested, an interesting example in the current context is the FIFO queue. Consider such a queue with unbounded capacity, with input port i and output port o. An infinite guarded recursive specification over the signature of the minimal theory MPT(A) is not hard to give. Suppose D is a finite data set, and notation of sequences over D is as before (see Section 5.6). The following recursive specification has a variable for each sequence over D.

$$
\begin{aligned}
Queue1 &= Q_\epsilon, \\
Q_\epsilon &= 1 + \sum_{d \in D} i?d.Q_d, \quad \text{and, for all } d \in D, \sigma \in D^*, \\
Q_{\sigma d} &= o!d.Q_\sigma + \sum_{e \in D} i?e.Q_{e\sigma d}.
\end{aligned}
$$

https://doi.org/10.1017/CBO9781139195003.011 Published online by Cambridge University Press

The last equation gives the behavior of the non-empty queue. Comparing the present specification with the one of the stack in Section 5.6 shows obvious similarity. In Section 6.6.2, it has been shown that theory TSP(A) allows a finite guarded recursive specification of the stack. Therefore, it may come as a surprise that there is no finite guarded recursive specification for the queue over the signature of theory TCP(A, γ) when communication is restricted to handshaking communication, which is an extension of the signature of TSP(A). The proof of this fact is outside the scope of this text. The interested reader is referred to (Baeten & Bergstra, 1988). Exercise 7.7.10 shows that the queue is finitely definable over TCP(A, γ) when arbitrary communication is allowed, showing that the restriction to handshaking in the above is essential.

Another interesting aspect is that some extensions of theory TCP(A, γ) are more expressive than the basic theory TCP(A, γ) itself when considering finite guarded recursion. It turns out that there is a finite guarded recursive specification for the queue in the basic theory $\mathrm{BCP}_{\mathrm{rec}}(A, \gamma)$ with handshaking communication *extended with renaming operators* (see Section 6.7), as the following shows. That is, the queue is finitely definable over theory (BCP + HA + RN)(A, γ). The specification assumes the standard communication function γ_S of Definition 7.3.1.

Consider two renaming functions $f, g : A \to A$. For all data elements $d \in D$, $f(o!d) = l!d$, and f leaves all other atomic actions unchanged; $g(l?d) = o!d$, for each $d \in D$, and g leaves all other atomic actions unchanged. Consider the following recursive specification with variables *Queue*2 and Z, with $H = \{l!d, l?d \mid d \in D\}$:

$$\begin{aligned} \mathit{Queue}2 &= 1 + \sum_{d \in D} i?d.\rho_g(\partial_H(\rho_f(\mathit{Queue}2) \parallel o!d.Z)), \\ Z &= 1 + \sum_{d \in D} l?d.Z. \end{aligned}$$

The proposition below proves that $((\mathrm{BCP} + \mathrm{HA} + \mathrm{RN})_{\mathrm{rec}} + \mathrm{RSP})(A, \gamma_S) \vdash \mathit{Queue}1 = \mathit{Queue}2$. In order to gain some more intuition about the specification of *Queue*2, consider the following. An unbounded queue can be imagined as a one-element buffer connected to an unbounded (sub)queue. Although this may seem strange at a first glance, it does work. The fact that the queue is unbounded is crucial. The first element put into the queue (through an $i?d$ action) is put into the one-place buffer; all following elements are put into the subqueue. The output port of the subqueue is some internal port l (which explains the renaming via function f of the recursive occurrence of *Queue*2 in the specification). This port is watched over by a guard Z, that becomes activated only if the one-element buffer is emptied (through an $o!d$ action). By encapsulating isolated communication actions over the internal port l and renaming success-

https://doi.org/10.1017/CBO9781139195003.011 Published online by Cambridge University Press

ful communications over this internal port to $o!d$ actions, this guard makes sure that the elements of the subqueue are forwarded to the outside world over port o as $o!d$ actions in the right order. Of course, the subqueue is itself again built from a one-element buffer and a subqueue ..., see Figure 7.6.

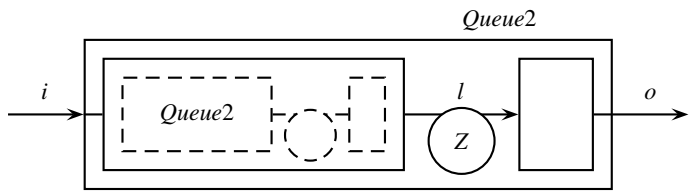

Fig. 7.6. The intuition behind the specification of *Queue*2.

Proposition 7.7.4 (Queues)

$$((\mathrm{BCP} + \mathrm{HA} + \mathrm{RN})_{\mathrm{rec}} + \mathrm{RSP})(A, \gamma_S) \vdash \mathit{Queue}1 = \mathit{Queue}2.$$

Proof By RSP, it is enough to show that process *Queue*2 is a solution of the recursive specification of *Queue*1. In order to do so, it is necessary to find an expression in terms of *Queue*2 for each variable Q_σ (with $\sigma \in D^*$) in the specification of *Queue*1. These expressions are defined inductively as follows.

Expressions R^n_σ, for each $n \in \mathbf{N}$ and each $\sigma \in D^*$, are defined by induction on n. So, first of all, the expressions R^0_σ are defined, by induction on σ:

$$\begin{aligned} R^0_\epsilon &= \mathit{Queue}2, \quad \text{and, for all } \sigma \in D^*, d \in D, \\ R^0_{\sigma d} &= \rho_g(\partial_H(\rho_f(R^0_\sigma) \parallel o!d.Z)). \end{aligned}$$

Next, given the expressions R^n_σ, for all $n \in \mathbf{N}$ and $\sigma \in D^*$, expressions R^{n+1}_σ are defined as follows:

$$R^{n+1}_\sigma = \rho_g(\partial_H(\rho_f(R^n_\sigma) \parallel Z)).$$

Using the equations in the specification for *Queue*2, the following can be derived, for each $n \in \mathbf{N}$ and each $d \in D, \sigma \in D^*$:

$$\begin{aligned} &(\mathrm{BCP} + \mathrm{HA} + \mathrm{RN})_{\mathrm{rec}}(A, \gamma_S) \vdash \\ &\qquad R^n_\epsilon = 1 + \sum_{d \in D} i?d.R^n_d, \\ &\qquad R^n_{\sigma d} = o!d.R^{n+1}_\sigma + \sum_{e \in D} i?e.R^n_{e\sigma d}. \end{aligned}$$

Following the reasoning in Example 5.5.13 (Recursion principle RSP), it can be inferred that $((\mathrm{BCP} + \mathrm{HA} + \mathrm{RN})_{\mathrm{rec}} + \mathrm{RSP})(A, \gamma_S) \vdash R^n_\sigma = Q_\sigma$ for all

https://doi.org/10.1017/CBO9781139195003.011 Published online by Cambridge University Press

$n \in \mathbf{N}, \sigma \in D^*$, and so in particular $((\mathrm{BCP} + \mathrm{HA} + \mathrm{RN})_{\mathrm{rec}} + \mathrm{RSP})(A, \gamma_S) \vdash Queue2 = R^0_\epsilon = Q_\epsilon = Queue1$. □

Exercises

7.7.1 Develop theory $\mathrm{TCP}(A, \gamma)$. That is, prove elimination and conservativity results, and give a term model proving soundness and ground-completeness.

7.7.2 Prove Theorem 7.7.1 (Standard concurrency).

7.7.3 Prove that the following identities can be derived from $\mathrm{TCP}(A, \gamma)$:

(a) $x \parallel 0 = x \cdot 0$;
(b) $x \lfloor\!\lfloor y \cdot 0 = x \cdot 0 \lfloor\!\lfloor y = (x \lfloor\!\lfloor y) \cdot 0$;
(c) $x \parallel y \cdot 0 = x \cdot 0 \parallel y = (x \parallel y) \cdot 0$.

7.7.4 Prove that, for all closed $\mathrm{TCP}(A, \emptyset)$-terms p and actions $a \in A$, $\mathrm{TCP}(A, \emptyset) \vdash p \cdot a.1 \parallel a.1 = (p \parallel a.1) \cdot a.1$.
(Note that this equality expresses a property about the free merge (i.e., without communication) and its interaction with sequential composition. It cannot be derived from the other axioms of $\mathrm{TCP}(A, \emptyset)$ for arbitrary open terms, not even in the presence of the Free-Merge Axiom FMA of Table 7.2. This is similar to the earlier results obtained for the Axioms of Standard Concurrency. A consequence of this result is that $(\mathrm{TCP} + \mathrm{FMA})(A, \emptyset)$ is not ω-complete (see Section 2.3).)

7.7.5 Consider Example 7.7.2 (CSP parallel composition and communication). Show that the given equality $b.a.b.a.1 \|^{\mathrm{CSP}}_S a.c.1 = b.a.(b.c.0 + c.b.0)$ can be derived from the theory $(\mathrm{TCP} + \mathrm{RN})(A, \gamma)$ extended with the defining axiom for CSP parallel composition.

7.7.6 Consider the so-called *laws* of CSP in (Hoare, 1985). Which of the laws concerning parallel composition can be derived in the theory $(\mathrm{TCP} + \mathrm{RN})(A, \gamma)$ extended with the defining axiom for CSP parallel composition in Example 7.7.2 (CSP parallel composition and communication)?

7.7.7 Consider a vending machine K, where coffee costs 35 cents. Three coins, of 20, 10, and 5 cents, need to be inserted (in arbitrary order) and then coffee is dispensed. Describe K by means of a recursive equation, using the merge operator without communication.

7.7.8 Sketch the transition system of the queue for the case $D = \{0, 1\}$.

7.7.9 Derive the recursive specification for the expressions R^n_σ given in the proof of Proposition 7.7.4 (Queues) from the equations of the specification for $Queue2$.

https://doi.org/10.1017/CBO9781139195003.011 Published online by Cambridge University Press

7.7.10 The queue is finitely definable over theory BCP(A, γ) when non-binary communication is allowed. Suppose there are actions $k(d)$, $m(d) \in A$ (for $d \in D$) such that $\gamma(o!d, k(d)) = m(d)$ and $\gamma(m(d), k(d)) = o!d$; let $H = \{o!d, m(d) \mid d \in D\}$ and $K = \{m(d), k(d) \mid d \in D\}$. Show that the queue can be defined by the following recursive specification:

$$\begin{aligned} Queue3 &= 1 + \sum_{d \in D} i?d.\partial_K(Z \parallel \partial_H(Queue3 \parallel m(d).Z)), \\ Z &= 1 + \sum_{d \in D} k(d).Z. \end{aligned}$$

7.8 Specifying the Alternating-Bit Protocol

This section presents a more elaborate example of an algebraic specification, namely the specification of a communication protocol. This protocol is often referred to as the Alternating-Bit Protocol in the literature. It concerns the transmission of data through an unreliable channel in such a way that – despite the unreliability – no information will get lost. The communication network used in the example is shown in Figure 7.7.

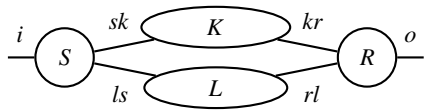

Fig. 7.7. The Alternating-Bit Protocol: communication network.

The following describes the components of this network. In Figure 7.7, S is the sender, sending data elements $d \in D$, with D a finite data domain, to the receiver R via the unreliable channel K. After having received a certain data element, R will send an acknowledgement to S via channel L which is unreliable as well. (In practice, K and L are usually physically the same medium.) The problem now is to define processes S and R such that no information will get lost; that is, the behavior of the entire process, apart from the communications at the internal ports sk, kr, rl, and ls, satisfies the equation

$$Buf1_{io} = 1 + \sum_{d \in D} i?d.o!d.Buf1_{io},$$

i.e., the process behaves externally as a one-place buffer with input port i and output port o (see Section 7.6).

A solution can be formulated as follows. The sender S reads a datum d at port i and passes on a sequence $d0, d0, d0, \ldots$ of copies of this datum with an appended bit 0 to K until an acknowledgement 0 is received at port ls. Then,

the next datum is read, and sent on together with a bit 1; the acknowledgement then is the reception of a 1. The following data element has, in turn, 0 as an appended bit. Thus, 0 and 1 form the alternating bit. A datum with an appended bit is called a *frame*.

The process K denotes the data transmission channel, passing on frames of the form $d0$ and $d1$. K may corrupt data, however, passing on $\bot$ (an error message; thus, it is assumed that the incorrect transmission of d can be recognized, for instance, using a checksum).

Receiver R gets frames $d0, d1$ from K, sending on d to port o (if this was not already done earlier), and sending the acknowledgement 0 resp. 1 to L.

The process L is the acknowledgement transmission channel, and passes bits 0 or 1, received from R, on to S. L is also unreliable, and may send on $\bot$ instead of 0 or 1.

The processes S, K, R, and L can be specified by means of recursive specifications. Let D be a finite data set, and define the set of frames by $F = \{d0, d1 \mid d \in D\}$. The following specification uses the standard communication function γ_S of Definition 7.3.1 (Standard communication), with $F \cup \{0, 1, \bot\}$ as the set of data elements.

Let t be some atomic action. The channels K and L are given by the following equations.

$$K = 1 + \sum_{x \in F} sk?x.(t.kr!x.K + t.kr!\bot.K),$$
$$L = 1 + \sum_{n \in \{0,1\}} rl?n.(t.ls!n.L + t.ls!\bot.L).$$

The actions t serve to make the choices non-deterministic: the decision whether or not the frame will be corrupted is internal to the channels, and cannot be influenced by the environment. The protocol to be specified is still functioning correctly, nonetheless, if the occurrences of internal action t are removed. Exercise 7.8.2 addresses this point in more detail.

The sender S and the receiver R are given by the following recursive specifications, for all $n \in \{0, 1\}$ and $d \in D$. Intuitively, Sn specifies the (possibly repeated) attempt to transmit a datum with appended bit n; Rn corresponds to the situation that the last properly received frame contained bit n, which means that the expected frame contains bit $1 - n$. The sender S iteratively executes behaviors $S0$ and $S1$, and the receiver R repeats $R1$ followed by $R0$.

$$\begin{aligned} S &= 1 + S0 \cdot S1 \cdot S, \\ Sn &= 1 + \sum_{d \in D} i?d.Sn_d, \\ Sn_d &= sk!dn.Tn_d, \\ Tn_d &= ls?(1-n).Sn_d + ls?\bot.Sn_d + ls?n.1, \end{aligned}$$

https://doi.org/10.1017/CBO9781139195003.011 Published online by Cambridge University Press

and

$$\begin{aligned} R &= 1 + R1 \cdot R0 \cdot R, \\ Rn &= 1 + kr?\bot.rl!n.Rn + \sum_{d \in D} kr?dn.rl!n.Rn \\ &\quad + \sum_{d \in D} kr?d(1-n).o!d.rl!(1-n).1. \end{aligned}$$

The composition of the four processes of the Alternating-Bit Protocol, enforcing communication, is represented by

$$\partial_H(S \parallel K \parallel L \parallel R),$$

where $H = \{p?x,\ p!x \mid x \in F \cup \{0, 1, \bot\},\ p \in \{sk, kr, rl, ls\}\}$.

The next step is to derive a recursive specification for this process. Consider the following recursive specification, using variables X, $X1_d$, $X2_d$, Y, $Y1_d$, and $Y2_d$ (for each $d \in D$).

$$\begin{aligned} X &= 1 + \sum_{d \in D} i?d.X1_d, \\ X1_d &= sk?d0.\,(t.kr?\bot.rl?1.(t.ls?\bot.X1_d + t.ls?1.X1_d) \\ &\quad + t.kr?d0.o!d.X2_d)\,, \\ X2_d &= rl?0.\,(t.ls?\bot.sk?d0.(t.kr?\bot.X2_d + t.kr?d0.X2_d) + t.ls?0.Y)\,, \\ Y &= 1 + \sum_{d \in D} i?d.Y1_d, \\ Y1_d &= sk?d1.\,(t.kr?\bot.rl?0.(t.ls?\bot.Y1_d + t.ls?0.Y1_d) \\ &\quad + t.kr?d1.o!d.Y2_d)\,, \\ Y2_d &= rl?1.\,(t.ls?\bot.sk?d1.(t.kr?\bot.Y2_d + t.kr?d1.Y2_d) + t.ls?1.X)\,. \end{aligned}$$

It can be proved that process $\partial_H(S \parallel K \parallel L \parallel R)$ is a solution of this recursive specification, i.e., that it satisfies the equation for X. The following list shows the processes that should be substituted for the variables in the above specification.

$$\begin{aligned} X &= \partial_H(S \parallel K \parallel L \parallel R), \\ X1_d &= \partial_H(S0_d \cdot S1 \cdot S \parallel K \parallel L \parallel R), \\ X2_d &= \partial_H(T0_d \cdot S1 \cdot S \parallel K \parallel L \parallel rl!0.R0 \cdot R), \\ Y &= \partial_H(S1 \cdot S \parallel K \parallel L \parallel R0 \cdot R), \\ Y1_d &= \partial_H(S1_d \cdot S \parallel K \parallel L \parallel R0 \cdot R), \\ Y2_d &= \partial_H(T1_d \cdot S \parallel K \parallel L \parallel rl!1.R). \end{aligned}$$

The expansion theorem for handshaking communication, Theorem 7.4.10, can be used to show that these processes form a solution. Part of the calculations are given in the following. In fact, these calculations show how the above specification and the process expressions to be substituted for the variables are *derived*. It is not necessary to 'invent' them beforehand.

The transition system for X is depicted in Figure 7.8. Here, the sum over different data elements in the nodes of X and Y is not shown, but just one,

https://doi.org/10.1017/CBO9781139195003.011 Published online by Cambridge University Press

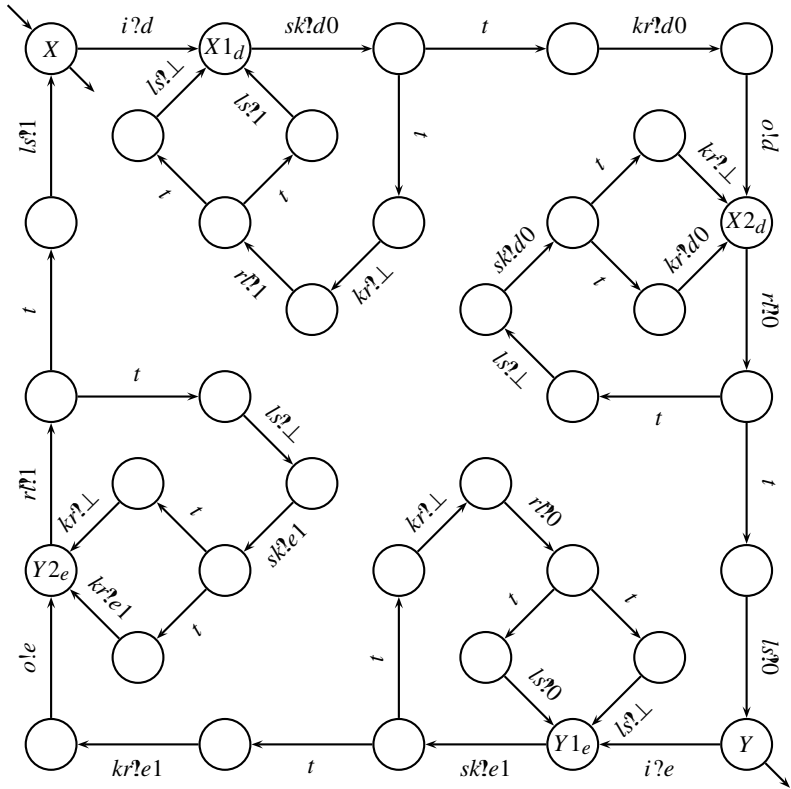

Fig. 7.8. Alternating-Bit Protocol.

arbitrary element is shown (d resp. e). To help the reader, variable names are given inside states.

For the first equation in the above list, observe that

$$\begin{aligned}(\mathrm{TCP} + \mathrm{HA})_{\mathrm{rec}}(A, \gamma_S) \vdash \\ \partial_H(S \parallel K \parallel L \parallel R) &= 1 + \sum_{d \in D} i?d.\partial_H(S0_d \cdot S1 \cdot S \parallel K \parallel L \parallel R) \\ &= 1 + \sum_{d \in D} i?d.X1_d.\end{aligned}$$

Note that the Handshaking Axiom is strictly speaking not necessary to arrive at the desired result. However, it is convenient in the derivation, because it allows to use the expansion theorem for handshaking, Theorem 7.4.10.

For the second equation, assume $d \in D$. In the following, two abbreviations are used, for $x \in F$ and $n \in \{0, 1\}$:

https://doi.org/10.1017/CBO9781139195003.011 Published online by Cambridge University Press

$$K'_x = t.kr!x.K + t.kr!\bot.K,$$
$$L'_n = t.ls!n.L + t.ls!\bot.L.$$

Consider

$$\begin{aligned}
&(\mathrm{TCP} + \mathrm{HA})_{\mathrm{rec}}(A, \gamma_S) \vdash \\
&\quad \partial_H(S0_d \cdot S1 \cdot S \parallel K \parallel L \parallel R) \\
&= sk?d0.\partial_H(T0_d \cdot S1 \cdot S \parallel K'_{d0} \parallel L \parallel R) \\
&= sk?d0.(t.\partial_H(T0_d \cdot S1 \cdot S \parallel kr!d0.K \parallel L \parallel R) \\
&\qquad + t.\partial_H(T0_d \cdot S1 \cdot S \parallel kr!\bot.K \parallel L \parallel R)) \\
&= sk?d0.(t.kr?d0.\partial_H(T0_d \cdot S1 \cdot S \parallel K \parallel L \parallel o!d.rl!0.R0 \cdot R) \\
&\qquad + t.kr?\bot.\partial_H(T0_d \cdot S1 \cdot S \parallel K \parallel L \parallel rl!1.R1 \cdot R0 \cdot R)) \\
&= sk?d0.(t.kr?d0.o!d.\partial_H(T0_d \cdot S1 \cdot S \parallel K \parallel L \parallel rl!0.R0 \cdot R) \\
&\qquad + t.kr?\bot.rl?1.\partial_H(T0_d \cdot S1 \cdot S \parallel K \parallel L'_1 \parallel R)) \\
&= sk?d0.(t.kr?d0.o!d.\partial_H(T0_d \cdot S1 \cdot S \parallel K \parallel L \parallel rl!0.R0 \cdot R) \\
&\qquad + t.kr?\bot.rl?1.(t.\partial_H(T0_d \cdot S1 \cdot S \parallel K \parallel ls!1.L \parallel R) \\
&\qquad\qquad + t.\partial_H(T0_d \cdot S1 \cdot S \parallel K \parallel ls!\bot.L \parallel R))) \\
&= sk?d0.(t.kr?d0.o!d.\partial_H(T0_d \cdot S1 \cdot S \parallel K \parallel L \parallel rl!0.R0 \cdot R) \\
&\qquad + t.kr?\bot.rl?1.(t.ls?1.\partial_H(S0_d \cdot S1 \cdot S \parallel K \parallel L \parallel R) \\
&\qquad\qquad + t.ls?\bot.\partial_H(S0_d \cdot S1 \cdot S \parallel K \parallel L \parallel R))).
\end{aligned}$$

The appropriate substitutions yield the second equation. Continuing with the third, assume again $d \in D$.

$$\begin{aligned}
&(\mathrm{TCP} + \mathrm{HA})_{\mathrm{rec}}(A, \gamma_S) \vdash \\
&\quad \partial_H(T0_d \cdot S1 \cdot S \parallel K \parallel L \parallel rl!0.R0 \cdot R) \\
&= rl?0.\partial_H(T0_d \cdot S1 \cdot S \parallel K \parallel L'_0 \parallel R0 \cdot R) \\
&= rl?0.(t.\partial_H(T0_d \cdot S1 \cdot S \parallel K \parallel ls!0.L \parallel R0 \cdot R) \\
&\qquad + t.\partial_H(T0_d \cdot S1 \cdot S \parallel K \parallel ls!\bot.L \parallel R0 \cdot R)) \\
&= rl?0.(t.ls?0.\partial_H(S1 \cdot S \parallel K \parallel L \parallel R0 \cdot R) \\
&\qquad + t.ls?\bot.\partial_H(S0_d \cdot S1 \cdot S \parallel K \parallel L \parallel R0 \cdot R)) \\
&= rl?0.(t.ls?0.\partial_H(S1 \cdot S \parallel K \parallel L \parallel R0 \cdot R) \\
&\qquad + t.ls?\bot.sk?d0.\partial_H(T0_d \cdot S1 \cdot S \parallel K'_{do} \parallel L \parallel R0 \cdot R)) \\
&= rl?0.(t.ls?0.\partial_H(S1 \cdot S \parallel K \parallel L \parallel R0 \cdot R) \\
&\qquad + t.ls?\bot.sk?d0. \\
&\qquad\qquad (t.\partial_H(T0_d \cdot S1 \cdot S \parallel kr!d0.K \parallel L \parallel R0 \cdot R) \\
&\qquad\qquad + t.\partial_H(T0_d \cdot S1 \cdot S \parallel kr!\bot.K \parallel L \parallel R0 \cdot R))) \\
&= rl?0.(t.ls?0.\partial_H(S1 \cdot S \parallel K \parallel L \parallel R0 \cdot R) \\
&\qquad + t.ls?\bot.sk?d0. \\
&\qquad\qquad (t.kr?d0.\partial_H(T0_d \cdot S1 \cdot S \parallel K \parallel L \parallel rl!0.R0 \cdot R) \\
&\qquad\qquad + t.kr?\bot.\partial_H(T0_d \cdot S1 \cdot S \parallel K \parallel L \parallel rl!0.R0 \cdot R))).
\end{aligned}$$

In this way, the first three equations have been obtained. The remaining three equations are dealt with in the same way.

https://doi.org/10.1017/CBO9781139195003.011 Published online by Cambridge University Press

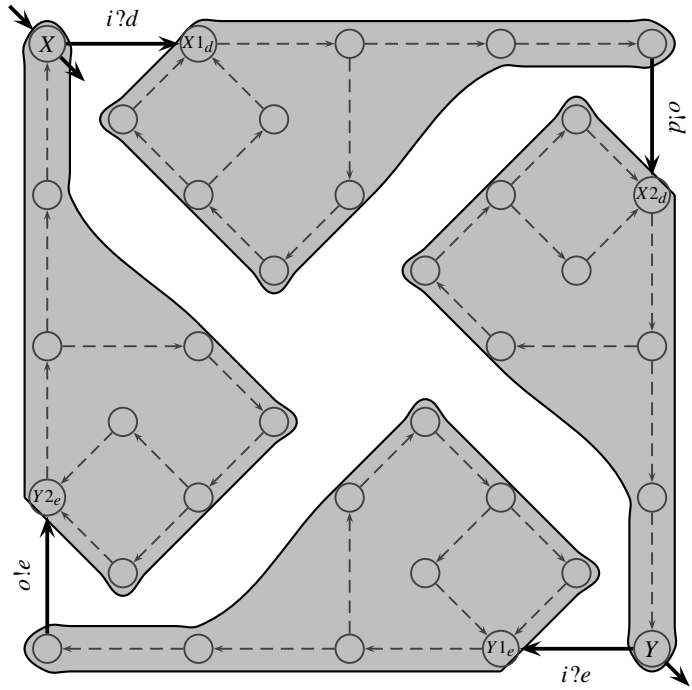

Fig. 7.9. Clustering of states in the Alternating-Bit Protocol.

Next, as indicated in the beginning of this section, it remains to be shown that process $\partial_H(S \parallel K \parallel L \parallel R)$, after abstraction of internal actions, satisfies the specification of the one-place buffer $Buf1_{io}$. The set of internal steps is $I = \{p?x \mid x \in F \cup \{0, 1, \bot\},\ p \in \{sk, kr, rl, ls\}\} \cup \{t\}$, i.e., all communications over internal ports and the internal actions t; only the actions $i?d$ and $o!d$, for any $d \in D$, that occur at external ports and are meant to communicate with the environment, are external. Thus, there is need of an abstraction operator τ_I, making internal steps in the parameter set I invisible, such that

$$\tau_I(\partial_H(S \parallel K \parallel L \parallel R)) = Buf1_{io}$$

is derivable from the theory extended with abstraction. Such a parameterized class of abstraction operators is defined in the following chapter. For now, a plausibility argument is given by means of a picture showing that, after abstraction, $\partial_H(S \parallel K \parallel L \parallel R)$ shows the intended behavior. In Figure 7.9, the shaded areas of the graph represent clusters, containing sets of internal states that are equivalent from the point of view of the environment.

https://doi.org/10.1017/CBO9781139195003.011 Published online by Cambridge University Press

Exercises

7.8.1 In this exercise, a simple communication protocol is considered. Data (from some finite data set D) are to be transmitted from a sender S to a receiver R through some unreliable channel K (see Figure 7.10).

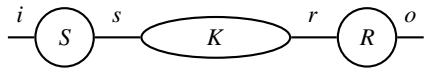

Fig. 7.10. A simple communication network.

The channel may forward the data correctly, or it may completely destroy data. The sender will send a data element until an acknowledgement *ack* is received. Consider the following specifications for S, K, and R:

$$\begin{aligned}
S &= 1 + \sum_{d \in D} i?d.S_d, \quad \text{with for all } d \in D,\\
S_d &= s!d.S_d + s?ack.S,\\
R &= 1 + \sum_{d \in D} r?d.o!d.R + r!ack.R,\\
K &= 1 + \sum_{d \in D} s?d.(t.K + t.r!d.L),\\
L &= r?ack.(t.L + t.s!ack.K).
\end{aligned}$$

In this specification, t is some internal action. Assume standard communication, and let $H = \{s!x, s?x, r!x, r?x \mid x \in D \cup \{ack\}\}$.

(a) Derive a recursive specification for the process $\partial_H(S \parallel K \parallel R)$, and draw the transition system.

(b) Does this communication protocol behave correctly?

7.8.2 Consider, in the Alternating-Bit Protocol, alternative channel specifications *without* the internal t-steps:

$$\begin{aligned}
Ka &= 1 + \sum_{x \in F} sk?x.(kr!x.Ka + kr!\bot.Ka),\\
La &= 1 + \sum_{n \in \{0,1\}} rl?n.(ls!n.La + ls!\bot.La).
\end{aligned}$$

Show that the receiver can *force* channel Ka to behave correctly, by refusing to accept errors, i.e., consider

$$Ra = \partial_{\{kr?\bot\}}(R)$$

and similarly

$$Sa = \partial_{\{ls?\bot\}}(S)$$

https://doi.org/10.1017/CBO9781139195003.011 Published online by Cambridge University Press

and show that in the process $\partial_H(Sa \parallel Ka \parallel La \parallel Ra)$ (with $H = \{p?x, p!x \mid x \in F \cup \{0, 1, \bot\},\ p \in \{sk, kr, rl, ls\}\}$ as above), errors $kr?\bot$ and $ls?\bot$ never occur, and that the protocol behaves correctly.
On the other hand, show that *with* the t actions, an occurrence of an error with the alternative sender and receiver specifications Sa and Ra results in deadlock, i.e., process $\partial_H(Sa \parallel K \parallel L \parallel Ra)$ has a deadlock (see Definition 4.4.14, adapted to the current setting).

7.9 Bibliographical remarks

The formulation of the theory TCP(A, γ) first appeared in (Baeten *et al.*, 2005). Earlier versions of TCP(A, γ) appear in (Baeten, 2003; Baeten & Reniers, 2004; Baeten & Bravetti, 2005); for older versions of similar theories with different names, see (Koymans & Vrancken, 1985; Vrancken, 1997; Baeten & Van Glabbeek, 1987). Replacing processes $a.1$ by constants a and removing action prefixing and the constant 1 yields the theory PA(A) of (Bergstra & Klop, 1982) when assuming a free merge (i.e., absence of communication, $\gamma = \emptyset$) and the theory ACP(A, γ) of (Bergstra & Klop, 1984a) for general communication functions. The theory MTCP(A, γ) corresponds to the theory of Basic Parallel Processes introduced in (Christensen, 1993) when the communication function γ is empty.

Communication and interaction occur in all concurrency theories. In (Hennessy, 1981), there is a communication function that has an identity element. Communication in CCS assumes both the presence of a so-called silent action τ (see the next chapter) and that all actions a have a so-called conjugate action $\bar{a}$. CCS communication is a special kind of handshaking with $\gamma(a, \bar{a}) = \tau$ for all a (see (Milner, 1980)). Since silent actions are related to abstraction; CCS communication combines these two concepts to some extent. In the current framework, communication and abstraction are separated, as discussed in more detail in the next chapter. CSP enforces communication of actions of the same name (see (Hoare, 1985)), a form of communication that is close to pure synchronization. Example 7.7.2 originates from (Baeten & Bravetti, 2006). In (Winskel, 1982), various communication formats are discussed.

The auxiliary operator $\between$ is from (Bergstra & Klop, 1982) (although something similar can be found in (Bekič, 1984)); the auxiliary operator $\mid$ was introduced in (Bergstra & Klop, 1984a). In (Hennessy, 1988b), a single auxiliary operator is used for an axiomatization of parallel composition.

The standard communication function is from (Bergstra & Klop, 1986c), here in a notation inspired by CSP. Example 7.4.1 (Communication) is from (Baeten & Weijland, 1990). The earliest version of the Axioms of Standard

Concurrency, the Handshaking Axiom and the expansion theorem can be found in (Bergstra & Tucker, 1984). The specification of the bag in one equation, and the proof that the bag has no finite guarded recursive specification over the signature of TSP(A), can be found in (Bergstra & Klop, 1984b). The specification of the queue in Section 7.7 is from (Baeten & Bergstra, 1988). It is owed to earlier work in (Bergstra & Tiuryn, 1987). For further work on queues, see (Van Glabbeek & Vaandrager, 1989; Denvir *et al.*, 1985; Broy, 1987; Hoare, 1985; Pratt, 1982). The Alternating-Bit Protocol of (Bartlett *et al.*, 1969) was verified in ACP-style process algebra in (Bergstra & Klop, 1986c). For further information, see (Koymans & Mulder, 1990; Van Glabbeek & Vaandrager, 1989; Larsen & Milner, 1987; Milner, 1989; Halpern & Zuck, 1987).