8
Abstraction

8.1 Introduction

In the previous chapter, Sections 7.6 and 7.8, the need arose to abstract from certain actions. Assume that the actions that need to be abstracted from are defined by a set $I \subseteq A$. Using skip operators ε_I (see Section 6.7) in some cases yields the desired result. With skipping, abstraction from an action b in the process described by $a.b.c.1$ is denoted as $\varepsilon_{\{b\}}(a.b.c.1)$. Using the previously given axioms (see Table 6.9), the result of this form of abstraction for this example is

$$\varepsilon_{\{b\}}(a.b.c.1) = a.c.1.$$

It does not always work out well to use a skip operator ε_I for the purpose of abstraction. This can be illustrated as follows. Consider the process described by the process term $a.1 + b.0$. Abstraction from the action b through the application of $\varepsilon_{\{b\}}$ to this process term results in

$$\varepsilon_{\{b\}}(a.1 + b.0) = a.1 + 0 = a.1.$$

The problem with this form of abstraction is that, before abstraction, it was evident that the process described has a deadlock whereas after abstraction the potential deadlock has disappeared. For practical use of the process theories in this book, a form of abstraction needs to be defined that does not lose so much information while abstracting from unimportant activity. This means that the abstraction mechanism should allow for the hiding of certain actions but it should not hide the consequences of the execution of those. An example of such a consequence is a potential deadlock.

This motivates the introduction of the silent step τ in this chapter. The silent step cannot be observed explicitly by an observer that is observing the behavior of a process. A silent step can be removed in some cases, but in other cases it cannot. In the process theories to be introduced, the silent step appears as

https://doi.org/10.1017/CBO9781139195003.012 Published online by Cambridge University Press

an action-prefix operator $\tau._-$. Abstraction, through abstraction operators τ_I (for $I \subseteq A$), then means the renaming of each occurrence of an action-prefix operation $a._-$ with $a \in I$ into a silent-step prefix operation.

The abstractions intended in the above examples can then be formulated as

$$\tau_{\{b\}}(a.b.c.1) = a.\tau.c.1$$

and

$$\tau_{\{b\}}(a.1 + b.0) = a.1 + \tau.0.$$

The occurrence of the silent step in the first example can be removed, because it cannot be observed by an external observer in any way, but in the second example it cannot be removed, because it can implicitly be observed via the deadlock resulting from its execution:

$$\tau_{\{b\}}(a.b.c.1) = a.\tau.c.1 = a.c.1$$

and

$$\tau_{\{b\}}(a.1 + b.0) = a.1 + \tau.0 \neq a.1 + 0 = a.1.$$

This chapter formally introduces abstraction and abstraction operators in the process-algebraic framework, and it is investigated when occurrences of silent steps are redundant and can hence be removed.

8.2 Transition systems with silent steps

The introduction of silent steps implies that it is necessary to reconsider the semantic framework used throughout this book. This section considers transition systems in which silent steps may occur. The resulting framework is an extension of the operational framework of Chapter 3. In the remainder, assume that $(S, L, \rightarrow, \downarrow)$ is a transition-system space, as defined in Definition 3.1.1. It is assumed that τ is a label, i.e., $\tau \in L$. Based on the observations in the introduction to this chapter, the following intuition can be formulated: if during the execution of a process a τ transition can be taken *without* discarding any of the options that were present before that transition, then this τ transition is redundant and can be removed. Figure 8.1 illustrates this situation. Observe that, assuming silent transitions are not externally visible, the process sketched on the left cannot be distinguished from the process on the right. After execution of the a action, the process on the left has the possibility of performing a step from x, or, after a silent step, a step from x or y. Assuming that an external observer cannot directly 'see' the silent step, this is equivalent to a process simply choosing between x and y as depicted on the right.

From the above discussion and examples, it should be clear that silent actions differ from other actions. This different behavior could potentially be

https://doi.org/10.1017/CBO9781139195003.012 Published online by Cambridge University Press

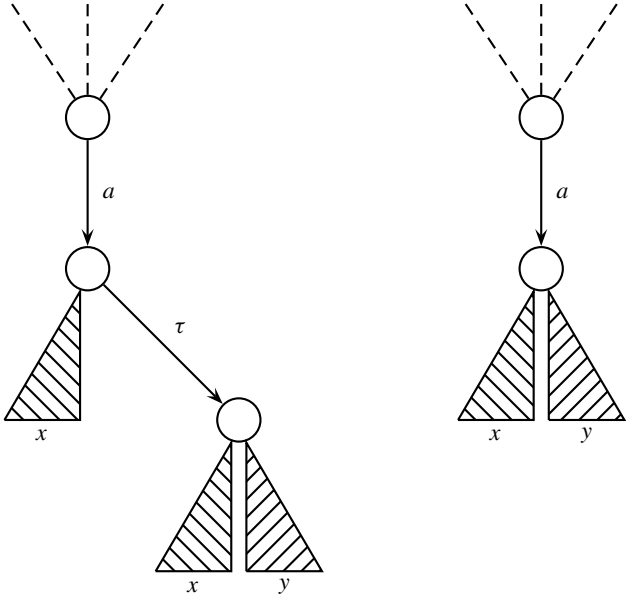


Fig. 8.1. Discarding silent steps.

realized by extra rules in a term deduction system that is used in the construction of a term model for some given process theory. However, a more generic and more elegant solution is to change the notion of equivalence on transition systems. The semantic equivalence in this chapter is no longer bisimilarity as considered before, but instead, (rooted) branching bisimilarity. In order to distinguish the various semantic equivalences, bisimilarity as considered up to this point, is sometimes called *strong* bisimilarity from now on.

To define branching bisimilarity, two auxiliary notions are needed. The following definition introduces the concept of reachability via silent steps.

Definition 8.2.1 (Reachable with τ-steps) The binary relation $\twoheadrightarrow$ on the states S of a transition-system space, denoting reachability via silent steps, is the smallest relation satisfying, for all states $s, t, u \in S$,

(i) $s \twoheadrightarrow s$;
(ii) whenever $s \xrightarrow{\tau} t$, then $s \twoheadrightarrow t$;
(iii) whenever $s \twoheadrightarrow t$ and $t \twoheadrightarrow u$, then $s \twoheadrightarrow u$.

Stated differently, relation $\twoheadrightarrow$ is the reflexive and transitive closure of the relation $\xrightarrow{\tau}$.

https://doi.org/10.1017/CBO9781139195003.012 Published online by Cambridge University Press

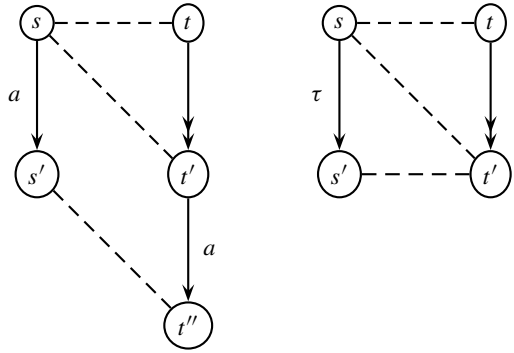


Fig. 8.2. Visualization of transfer condition (i) of a branching bisimulation.

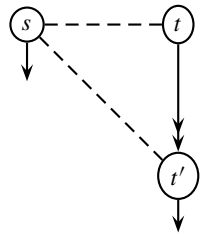


Fig. 8.3. Visualization of transfer condition (iii) of a branching bisimulation.

The following notational abbreviation is introduced to allow shorter formulations.

Notation 8.2.2 Let for any states $s, t \in S$ of a transition-system space and action $a \in L$, notation $s \xrightarrow{(a)} t$ be an abbreviation of $s \xrightarrow{a} t$, or $a = \tau$ and $s = t$. That is, $s \xrightarrow{(\tau)} t$ means that either one or zero τ-steps are performed; $s \xrightarrow{(a)} t$ with $a \neq \tau$ means that a single a-step is performed.

Definition 8.2.3 (Branching bisimilarity) A binary relation R on the set of states S of a transition-system space is a *branching bisimulation* relation if and only if the following so-called transfer conditions hold:

(i) for all states $s, t, s' \in S$, whenever $(s, t) \in R$ and $s \xrightarrow{a} s'$ for some $a \in L$, then there are states t' and t'' in S such that $t \twoheadrightarrow t'$ and $t' \xrightarrow{(a)} t''$ and both $(s, t') \in R$ and $(s', t'') \in R$;

(ii) vice versa, for all states $s, t, t' \in S$, whenever $(s, t) \in R$ and $t \xrightarrow{a} t'$ for some $a \in L$, then there are states s' and s'' in S such that $s \twoheadrightarrow s'$ and $s' \xrightarrow{(a)} s''$ and both $(s', t) \in R$ and $(s'', t') \in R$;

https://doi.org/10.1017/CBO9781139195003.012 Published online by Cambridge University Press

(iii) whenever $(s, t) \in R$ and $s\downarrow$, then there is a state $t' \in S$ such that $t \twoheadrightarrow t'$, $t'\downarrow$, and $(s, t') \in R$;

(iv) whenever $(s, t) \in R$ and $t\downarrow$, then there is a state $s' \in S$ such that $s \twoheadrightarrow s'$, $s'\downarrow$, and $(s', t) \in R$.

The first and third conditions are visualized in Figures 8.2 and 8.3. The branching bisimulation relations are indicated by the dashed lines connecting the states.

Two transition systems $s, t \in S$ are *branching bisimulation equivalent* or *branching bisimilar*, notation $s \leftrightarrows_{\mathrm{b}} t$, if and only if there is a branching bisimulation relation R on S with $(s, t) \in R$.

Example 8.2.4 (Branching bisimilarity) In Figure 8.4, two pairs of branching bisimilar transition systems are depicted.

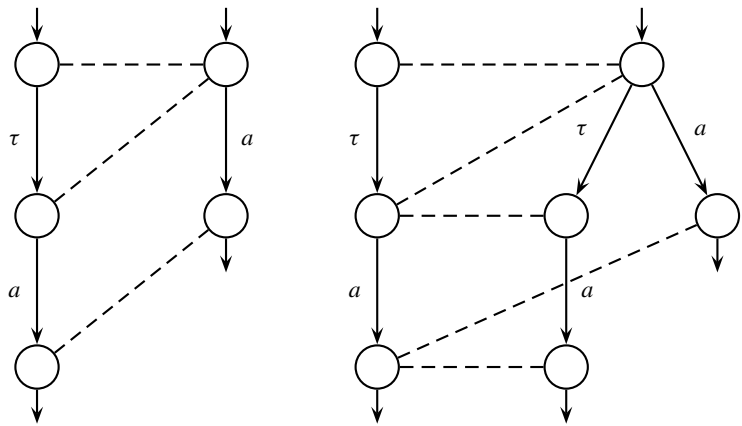


Fig. 8.4. Examples of branching bisimilar transition systems.

Theorem 8.2.5 (Equivalence) Branching bisimilarity is an equivalence.

Proof Let S be the set of states of a transition-system space. First, the relation $R = \{(s, s) \mid s \in S\}$ is obviously a branching bisimulation relation. This proves that $s \leftrightarrows_{\mathrm{b}} s$ for any transition system induced by any state $s \in S$, showing reflexivity of branching bisimilarity. Second, assume that $s \leftrightarrows_{\mathrm{b}} t$ for states $s, t \in S$. If R is a branching bisimulation relation witnessing $s \leftrightarrows_{\mathrm{b}} t$, then the relation $R' = \{(v, u) \mid (u, v) \in R\}$ is a branching bisimulation relation as well. This implies that $t \leftrightarrows_{\mathrm{b}} s$, showing symmetry of branching bisimilarity. Third, assume that $s \leftrightarrows_{\mathrm{b}} t$ and $t \leftrightarrows_{\mathrm{b}} u$ for states $s, t, u \in S$. Let

https://doi.org/10.1017/CBO9781139195003.012 Published online by Cambridge University Press

R_1 and R_2 be branching bisimulation relations witnessing $s \leftrightarrows_b t$ and $t \leftrightarrows_b u$, respectively. Then the relation composition $R_1 \circ R_2$ is a branching bisimulation relation witnessing $s \leftrightarrows_b u$, showing transitivity and completing the proof. The interested reader is referred to (Basten, 1998) for a more detailed proof. □

In order to define a process theory for a semantic equivalence, it is important that such an equivalence is a congruence with respect to the operators that are in the signature of that equational theory. (In other words, the equivalence should be a congruence on the term algebra.) This also applies to branching bisimilarity. So, to define a process theory with silent actions that extends the process theory BSP(A), it is required that the equivalence to be axiomatized is a congruence for the action-prefix operators and alternative composition.

It turns out that the notion of branching bisimilarity is lacking in this respect, as it is not a congruence with respect to alternative composition, assuming that the deduction rules for the alternative-composition operator remain as they are (see Table 4.2). Consider the process terms $a.1$ and $\tau.a.1$. To these process terms, the transition systems from Figure 8.5 can be associated. A branching

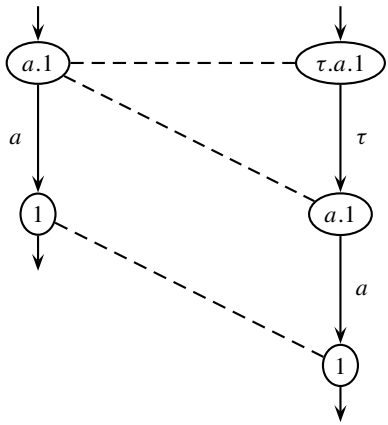


Fig. 8.5. Transition systems $a.1$ and $\tau.a.1$.

bisimulation relation on the states is drawn in the figure as well. Hence, the states represented by these two process terms are branching bisimilar. (Note the correspondence between the transition systems of Figure 8.5 and the two leftmost transition systems in Figure 8.4.) Now consider the transition systems associated to the process terms $a.1 + b.1$ and $\tau.a.1 + b.1$ in Figure 8.6. If branching bisimilarity would be a congruence with respect to alternative composition, then the states represented by these two process terms would be branching bisimilar as well. This is not the case. By definition, the two

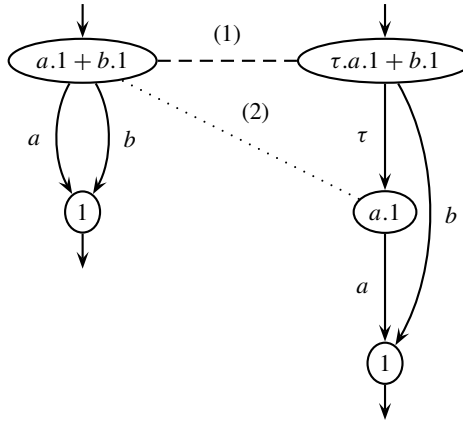


Fig. 8.6. Transition systems $a.1 + b.1$ and $\tau.a.1 + b.1$.

initial states of the transition systems should be related (1). As the right initial state has a τ transition to the state $a.1$, necessarily (by transfer condition (ii) of Definition 8.2.3 (Branching bisimilarity)) this state must be related to the initial state of the left transition system (2). But, this cannot be, as the one only allows an a transition, whereas the other one also allows a b transition.

Thus, the notion of branching bisimilarity turns out not to be a congruence with respect to the alternative composition and it will be necessary to define an extra condition, by which a notion that is a congruence can be formulated. The extra condition should ensure that processes $a.1$ and $\tau.a.1$ are considered different.

It is possible to give an intuitive explanation as to why pairs of processes like $a.1$ and $\tau.a.1$ should be considered different. Assume that a machine performing atomic actions is observed and that the beginning of an atomic step is observable. With this interpretation, the silent step can be interpreted as some action of which also the beginning cannot be observed. It follows from this intuition that the processes $a.1$ and $\tau.a.1$ are not equivalent, whereas they are branching bisimilar. The one process will immediately start by performing a whereas the other process will wait some time before showing its first observable action.

The extra condition that is formulated next amounts to saying that the pair of initial states should be treated differently, when relating them by a branching bisimulation: the initial step of a process cannot be matched by a step of the other process that has preceding silent steps, and the relation is therefore like a strong bisimulation initially.

Definition 8.2.6 (Rooted branching bisimilarity) Two transition systems s and t in S are *rooted branching bisimulation equivalent* or *rooted branching bisimilar*, notation $s \leftrightarrows_{\mathrm{rb}} t$, if and only if there is a branching bisimulation relation R on S with $(s, t) \in R$ such that

(i) for all states $s' \in S$, whenever $s \xrightarrow{a} s'$ for some $a \in L$, then there is a state t' such that $t \xrightarrow{a} t'$ and $(s', t') \in R$;
(ii) vice versa, for all states $t' \in S$, whenever $t \xrightarrow{a} t'$ for some $a \in L$, then there is a state s' such that $s \xrightarrow{a} s'$ and $(s', t') \in R$;
(iii) whenever $s\downarrow$, then $t\downarrow$;
(iv) whenever $t\downarrow$, then $s\downarrow$.

Usually, a branching bisimulation relation that satisfies the above so-called root conditions with respect to the states s and t is called a rooted branching bisimulation relation for s and t.

Example 8.2.7 (Rooted branching bisimilarity) The two transition systems of Figure 8.5 are not rooted branching bisimilar; in particular, the branching bisimulation given in Figure 8.5 is not a rooted branching bisimulation.

Figure 8.7 depicts two pairs of rooted branching bisimilar transition systems. The rooted branching bisimulation relations are as usual indicated by the dashed lines. In Figure 8.8, a rooted branching bisimulation relation is drawn for two slightly more complex transition systems.

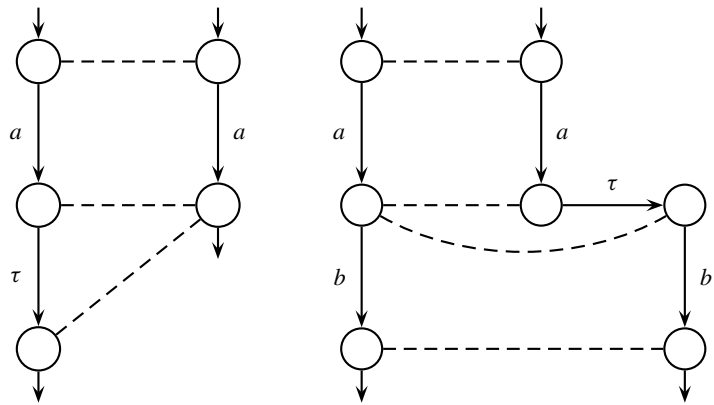


Fig. 8.7. Examples of rooted branching bisimilar transition systems.

Before, it was shown that branching bisimilarity is not a congruence with respect to alternative composition by means of a counterexample. For rooted branching bisimilarity, this is not a counterexample since the transition systems from Figure 8.5 are not rooted branching bisimilar.

https://doi.org/10.1017/CBO9781139195003.012 Published online by Cambridge University Press

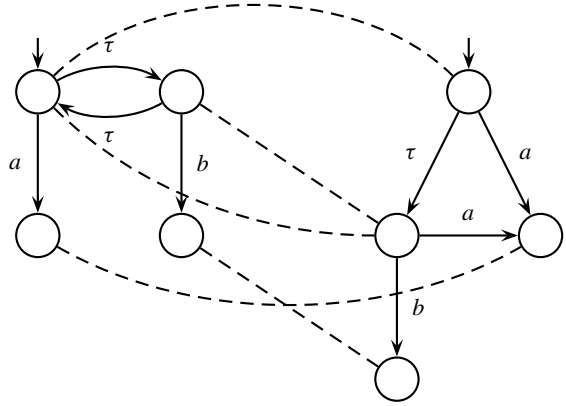


Fig. 8.8. Another example of rooted branching bisimilar transition systems.

Rooted branching bisimilarity turns out to be an equivalence relation.

Theorem 8.2.8 (Equivalence) Rooted branching bisimilarity is an equivalence relation.

Proof The relations that prove that branching bisimilarity is an equivalence also prove that rooted branching bisimilarity is an equivalence. □

Theorem 8.2.9 (Relation between bisimilarity notions) The following relations hold between the three semantic equivalences introduced so far in Chapter 3 and this chapter:

$$\leftrightarrows \subseteq \leftrightarrows_{\mathrm{rb}} \subseteq \leftrightarrows_{\mathrm{b}}.$$

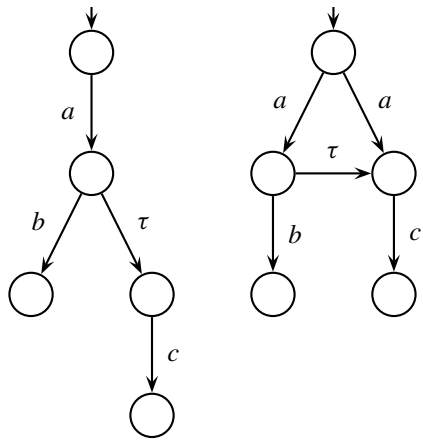
Proof Suppose that $p \leftrightarrows q$, i.e., p and q are strongly bisimilar. By definition this means that there exists a strong bisimulation relation R such that $(p, q) \in R$. Using the facts that $s \xrightarrow{a} t$ implies $s \xrightarrow{(a)} t$ and that $t \twoheadrightarrow t$ always holds, every strong bisimulation relation is also a branching bisimulation relation. Equally obvious is that every strong bisimulation relation R satisfies the root conditions for each of its elements. Therefore, the strong bisimulation relation R witnessing the bisimilarity of p and q is also a rooted branching bisimulation relation for p and q. This proves $\leftrightarrows \subseteq \leftrightarrows_{\mathrm{rb}}$.
Suppose that $p \leftrightarrows_{\mathrm{rb}} q$, i.e., p and q are rooted branching bisimilar. Then, by definition, there is a rooted branching bisimulation relation R such that $(p, q) \in R$. Since R is also a branching bisimulation, p and q are branching bisimilar: $p \leftrightarrows_{\mathrm{b}} q$. This proves that $\leftrightarrows_{\mathrm{rb}} \subseteq \leftrightarrows_{\mathrm{b}}$. □

https://doi.org/10.1017/CBO9781139195003.012 Published online by Cambridge University Press

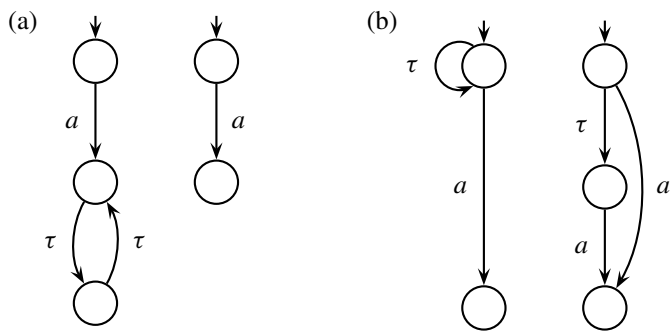
The theorem cannot in general be strengthened. For transition-system spaces without silent steps, the three semantic equivalences are equal, so none of the subset relations in the theorem is strict in general. However, the examples involving silent steps given so far show that the three semantic equivalences are in general all different, in the sense that they relate different transition systems for most non-trivial transition-system spaces with silent actions, so none of the subset relations in the theorem are equalities in general.

Exercises

8.2.1 Are the following transition systems branching bisimilar? If so, give a branching bisimulation relation between the two systems; otherwise, explain why they are not branching bisimilar.

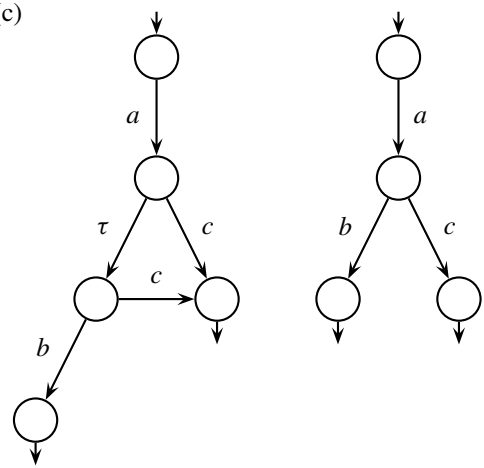


8.2.2 Are the following pairs of transition systems rooted branching bisimilar? If so, give a rooted branching bisimulation relation between the two systems; otherwise, explain why they are not rooted branching bisimilar.



https://doi.org/10.1017/CBO9781139195003.012 Published online by Cambridge University Press

(c)



8.2.3 Starting from Exercise 3.1.9, try to develop a notion of coloring for branching bisimilarity. That is, develop the concepts of abstract colored traces and coloring schemes that precisely capture branching bisimilarity and rooted branching bisimilarity.

8.2.4 A binary relation R on the set of states S of a transition-system space is called a *weak bisimulation* relation if and only if the following conditions hold:

(a) for all states $s, t, s' \in S$, whenever $(s, t) \in R$ and $s \xrightarrow{a} s'$ for some $a \in L$, then there are states t_1, t_2, t' in S such that $t \twoheadrightarrow t_1$, $t_1 \xrightarrow{(a)} t_2$ and $t_2 \twoheadrightarrow t'$, such that $(s', t') \in R$;

(b) vice versa, for all states $s, t, t' \in S$, whenever $(s, t) \in R$ and $t \xrightarrow{a} t'$ for some $a \in L$, then there are states s_1, s_2, s' in S such that $s \twoheadrightarrow s_1$, $s_1 \xrightarrow{(a)} s_2$ and $s_2 \twoheadrightarrow s'$ such that $(s', t') \in R$;

(c) whenever $(s, t) \in R$ and $s\downarrow$, then there is a state $t' \in S$ such that $t \twoheadrightarrow t'$ and $t'\downarrow$;

(d) whenever $(s, t) \in R$ and $t\downarrow$, then there is a state $s' \in S$ such that $s \twoheadrightarrow s'$ and $s'\downarrow$.

Prove that weak bisimilarity is an equivalence relation on the set of states S. Prove that, whenever two states are branching bisimilar, they are also weakly bisimilar. Give an example of two transition systems that are weakly bisimilar but not branching bisimilar.

8.2.5 Two transition systems s and t in S are *rooted weak bisimulation equivalent* or *rooted weakly bisimilar* if and only if there is a weak bisimulation relation R on S with $(s, t) \in R$ such that

https://doi.org/10.1017/CBO9781139195003.012 Published online by Cambridge University Press

(a) for all states $s' \in S$, whenever $s \xrightarrow{a} s'$ for some $a \in L, a \neq \tau$, then there is a state t' such that $t \xrightarrow{a} t'$ and $(s', t') \in R$;
(b) vice versa, for all states $t' \in S$, whenever $t \xrightarrow{a} t'$ for some $a \in L, a \neq \tau$, then there is a state s' such that $s \xrightarrow{a} s'$ and $(s', t') \in R$;
(c) whenever $s\downarrow$, then $t\downarrow$;
(d) whenever $t\downarrow$, then $s\downarrow$.

Prove that rooted weak bisimilarity is an equivalence relation on S. Prove that, whenever two states are rooted branching bisimilar, they are also rooted weakly bisimilar. Give an example of two transition systems that are rooted weakly bisimilar but not branching bisimilar.

8.3 BSP with silent steps

This section introduces the process theory $\mathrm{BSP}_\tau(A)$. This theory is the extension of $\mathrm{BSP}(A)$ with the previously mentioned τ-prefix operator $\tau._-$. The axioms of the process theory $\mathrm{BSP}_\tau(A)$ are the axioms of the process theory $\mathrm{BSP}(A)$ (given in Table 4.3) and the axiom given in Table 8.1. All atomic actions a that occur in the axioms are from now on assumed to be from the set $A_\tau = A \cup \{\tau\}$, unless explicitly stated otherwise.

$\mathrm{BSP}_\tau(A)$	
$\mathrm{BSP}(A)$;	
unary: $\tau._-$;	
x, y;	
$a.(\tau.(x + y) + x) = a.(x + y)$	B

Table 8.1. The process theory $\mathrm{BSP}_\tau(A)$ (with $a \in A_\tau$).

The one and only new axiom in $\mathrm{BSP}_\tau(A)$ is Axiom B, the so-called *Branching Axiom*. It captures precisely which occurrences of the silent step are without consequences and can therefore be omitted. This is illustrated by means of the schematic representation of the Branching Axiom in Figure 8.9. The execution of the internal step τ only adds alternatives. In other words, the alternatives that could have been chosen instead of executing the internal step, are still alternatives once this internal step has been executed.

Example 8.3.1 (A derivation) A derivation of

$$\mathrm{BSP}_\tau(A) \vdash c.(\tau.(b.1 + a.1) + \tau.(a.1 + b.1)) = c.(a.1 + b.1)$$

using Axiom B can be given as follows:

https://doi.org/10.1017/CBO9781139195003.012 Published online by Cambridge University Press

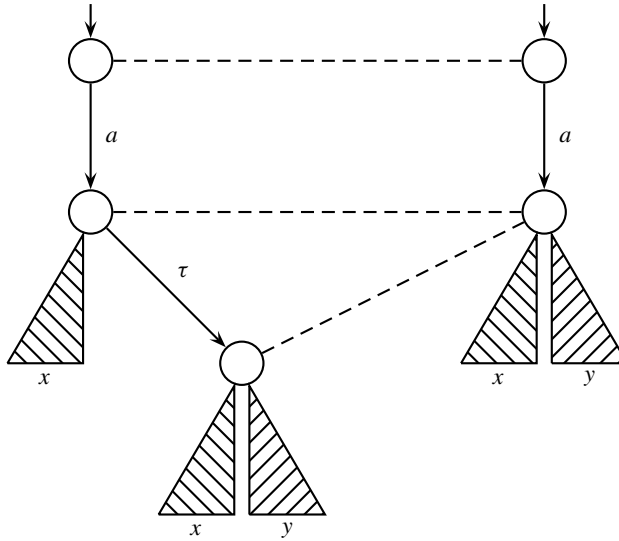


Fig. 8.9. Schematic representation of the Branching Axiom.

$$\begin{aligned}\mathrm{BSP}_\tau(A) \vdash c.(\tau.(b.1 + a.1) + \tau.(a.1 + b.1)) \\ = c.(\tau.(a.1 + b.1) + \tau.(a.1 + b.1)) \\ = c.(\tau.(a.1 + b.1)) = c.(\tau.(0 + a.1 + b.1) + 0) \\ = c.(0 + a.1 + b.1) = c.(a.1 + b.1).\end{aligned}$$

For obvious reasons, it is impossible to obtain an elimination theorem, i.e., it is impossible to reduce an arbitrary closed $\mathrm{BSP}_\tau(A)$-term to a $\mathrm{BSP}(A)$-term. Nevertheless, it can be shown that $\mathrm{BSP}_\tau(A)$ is a conservative ground-extension of $\mathrm{BSP}(A)$.

Theorem 8.3.2 (Conservative ground-extension) Process theory $\mathrm{BSP}_\tau(A)$ is a conservative ground-extension of process theory $\mathrm{BSP}(A)$.

Proof Exercise 8.3.4. □

Exercises

8.3.1 Prove that $\mathrm{BSP}_\tau(A) \vdash a.(\tau.b.1 + b.1) = a.\tau.(\tau.b.1 + \tau.\tau.b.1)$.

8.3.2 Prove that $\mathrm{BSP}_\tau(A) \vdash a.\tau.x = a.x$ for all $a \in A_\tau$.

8.3.3 Prove that $\mathrm{BSP}_\tau(A) \vdash a.(\tau.x + x) = a.x$ for all $a \in A_\tau$.

8.3.4 Prove Theorem 8.3.2 (Conservative ground-extension).

https://doi.org/10.1017/CBO9781139195003.012 Published online by Cambridge University Press

(Hint: construct a proof based on a term rewriting system, as illustrated in the proof of Theorem 4.5.3 (Conservative ground-extension of projection). The term rewriting system needs three variants of Axiom B. A proof via the results developed in Section 3.2 is not possible since this meta-theory assumes strong bisimilarity as a semantic equivalence, and therefore does not apply to the current setting with (rooted) branching bisimilarity as the equivalence notion.)

8.4 The term model

A term model for the process theory $\mathrm{BSP}_\tau(A)$ can be developed along the same lines as in earlier chapters. First, the term algebra $\mathbb{P}(\mathrm{BSP}_\tau(A))$ is defined.

Definition 8.4.1 (Term algebra) The *term algebra* for $\mathrm{BSP}_\tau(A)$ is the algebra $\mathbb{P}(\mathrm{BSP}_\tau(A)) = (\mathcal{C}(\mathrm{BSP}_\tau(A)), +, (a._-)_{a\in A}, \tau._-, 0, 1)$.

The term deduction system for $\mathrm{BSP}_\tau(A)$ consists of the same deduction rules as the deduction rules for $\mathrm{BSP}(A)$ of Tables 4.2 and 4.4 with the only difference that now $a \in A_\tau$.

As already mentioned in Section 8.2, rooted branching bisimilarity is an equivalence. It is not hard to prove that it is also a congruence on the term algebra for $\mathrm{BSP}_\tau(A)$. Recall that branching bisimilarity is not a congruence with respect to alternative composition. Note that it is not possible to prove the desired result via Theorem 3.2.7 (Congruence theorem), because that only applies to strong bisimilarity and not to rooted branching bisimilarity.

Theorem 8.4.2 (Congruence) Rooted branching bisimilarity is a congruence on the term algebra $\mathbb{P}(\mathrm{BSP}_\tau(A))$.

Proof To prove that rooted branching bisimilarity is a congruence on $\mathbb{P}(\mathrm{BSP}_\tau(A))$, it is necessary to show that for each n-ary ($n \geq 1$) function f of $\mathbb{P}(\mathrm{BSP}_\tau(A))$ and for all $p_1, \ldots, p_n, q_1, \ldots, q_n \in \mathcal{C}(\mathrm{BSP}_\tau(A))$, $p_1 \leftrightarrows_{\mathrm{rb}} q_1, \cdots, p_n \leftrightarrows_{\mathrm{rb}} q_n$ implies that $f(p_1, \ldots, p_n) \leftrightarrows_{\mathrm{rb}} f(q_1, \ldots, q_n)$.

(i) Alternative composition. Suppose that $p_1 \leftrightarrows_{\mathrm{rb}} q_1$ and $p_2 \leftrightarrows_{\mathrm{rb}} q_2$. By definition, this means that there exist branching bisimulation relations R_1 and R_2 such that the pairs $(p_1, q_1) \in R_1$ and $(p_2, q_2) \in R_2$ satisfy the root conditions of Definition 8.2.6 (Rooted branching bisimilarity). It needs to be shown that $p_1 + p_2 \leftrightarrows_{\mathrm{rb}} q_1 + q_2$.
Define $R = R_1 \cup R_2 \cup \{(p_1 + p_2, q_1 + q_2)\}$. The pairs in R that are

https://doi.org/10.1017/CBO9781139195003.012 Published online by Cambridge University Press

also in R_1 or in R_2 trivially satisfy the transfer conditions of Definition 8.2.3 (Branching bisimilarity). Thus, it remains to check whether the pair $(p_1 + p_2, q_1 + q_2) \in R$ satisfies the transfer conditions of Definition 8.2.3 and the root conditions of Definition 8.2.6 (Rooted branching bisimilarity). Since the root conditions imply the transfer conditions, in effect, this means that only the root conditions have to be checked.

First, consider root condition (i) of Definition 8.2.6. Suppose that $p_1 + p_2 \stackrel{a}{\rightarrow} r_1$ for some $a \in A_\tau$ and $r_1 \in \mathcal{C}(\mathrm{BSP}_\tau(A))$. Then, by the deduction rules of Table 4.2, $p_1 \stackrel{a}{\rightarrow} r_1$ or $p_2 \stackrel{a}{\rightarrow} r_1$. The two cases are similar; hence only the first case is elaborated. From the assumptions that $p_1 \leftrightarrows_{\mathrm{rb}} q_1$ and that this rooted branching bisimilarity relationship is witnessed by the relation R_1, it follows that $q_1 \stackrel{a}{\rightarrow} s_1$ for some s_1 in $\mathcal{C}(\mathrm{BSP}_\tau(A))$ such that $(r_1, s_1) \in R_1$. Using the deduction rules of Table 4.2 then also $q_1 + q_2 \stackrel{a}{\rightarrow} s_1$. Furthermore, as $R_1 \subseteq R$, $(r_1, s_1) \in R$, this completes the proof for condition (i).

The proof for root condition (ii) is omitted as it is similar to the proof for condition (i).

Consider root condition (iii). Suppose that $(p_1 + p_2)\downarrow$. Then $p_1 \downarrow$ or $p_2 \downarrow$. From the assumption that $p_1 \leftrightarrows_{\mathrm{rb}} q_1$ and $p_2 \leftrightarrows_{\mathrm{rb}} q_2$, it follows that $q_1 \downarrow$ or $q_2 \downarrow$. Thus, in either case $(q_1 + q_2)\downarrow$.

The proof for root condition (iv) is again omitted as it is similar to the previous proof.

(ii) Action prefix. Suppose that $p \leftrightarrows_{\mathrm{rb}} q$ is witnessed by the rooted branching bisimulation relation R. Define $R' = R \cup \{(a.p, a.q)\}$. It has to be shown that R' is a branching bisimulation and that the pair $(a.p, a.q)$ satisfies the root conditions.

As R is assumed to be a branching bisimulation, all pairs in R' that are also in R satisfy the transfer conditions. Hence, it remains to check whether the pair $(a.p, a.q)$ satisfies the transfer conditions. Since in addition for $(a.p, a.q)$ the root conditions must be satisfied, it suffices to establish only the root conditions. Both $a.p$ and $a.q$ cannot terminate: $(a.p)\not\downarrow$ and $(a.q)\not\downarrow$. Each of these process terms can perform an a transition only: $a.p \stackrel{a}{\rightarrow} p$ and $a.q \stackrel{a}{\rightarrow} q$. It must be shown that the resulting states (process terms, i.e., p and q) are related by R'. As R is a (rooted) branching bisimulation relation witnessing $p \leftrightarrows_{\mathrm{rb}} q$, by definition $(p, q) \in R$. As $R \subseteq R'$, then also $(p, q) \in R'$.

(iii) τ-prefix. Suppose that $p \leftrightarrows_{\mathrm{rb}} q$ is witnessed by the rooted branching bisimulation relation R. Define $R' = R \cup \{(\tau.p, \tau.q)\}$. It has to be shown that R' is a branching bisimulation relation and that $(\tau.p, \tau.q)$

https://doi.org/10.1017/CBO9781139195003.012 Published online by Cambridge University Press

satisfies the root conditions. Again, proving that $(\tau.p, \tau.q)$ satisfies the root conditions suffices. This part is trivial and therefore omitted.

□

The term model of $\mathrm{BSP}_\tau(A)$ is obtained in the standard way by considering the rooted branching bisimilarity quotient of the term algebra. It can be shown that $\mathrm{BSP}_\tau(A)$ is indeed a sound axiomatization of rooted branching bisimilarity on closed process terms. It turns out that it is also ground-complete.

Definition 8.4.3 (Term model of $\mathrm{BSP}_\tau(A)$) The term model of process theory $\mathrm{BSP}_\tau(A)$ is the quotient algebra $\mathbb{P}(\mathrm{BSP}_\tau(A))_{/\leftrightarrows_{\mathrm{rb}}}$, with $\mathbb{P}(\mathrm{BSP}_\tau(A))$ the term algebra of Definition 8.4.1.

Theorem 8.4.4 (Soundness) Theory $\mathrm{BSP}_\tau(A)$ is a sound axiomatization of algebra $\mathbb{P}(\mathrm{BSP}_\tau(A))_{/\leftrightarrows_{\mathrm{rb}}}$, i.e., $\mathbb{P}(\mathrm{BSP}_\tau(A))_{/\leftrightarrows_{\mathrm{rb}}} \models \mathrm{BSP}_\tau(A)$.

Proof It must be shown that, for each axiom $s = t$ of $\mathrm{BSP}_\tau(A)$, $\mathbb{P}(\mathrm{BSP}_\tau(A))_{/\leftrightarrows_{\mathrm{rb}}} \models s = t$. Hence, for each axiom $s = t$ it must be shown that $p \leftrightarrows_{\mathrm{rb}} q$ for any closed instantiation $p = q$ of $s = t$.
For each axiom $s = t$ of $\mathrm{BSP}_\tau(A)$ that is also an axiom of $\mathrm{BSP}(A)$, it has already been shown that $p \leftrightarrows q$ for any closed instantiation $p = q$ of $s = t$; see the proofs of Theorems 4.3.5 (Soundness of $\mathrm{MPT}(A)$) and 4.4.7 (Soundness of $\mathrm{BSP}(A)$). Note that these proofs carry over to the current setting where closed terms may contain τ-prefix operators. As $\leftrightarrows \subseteq \leftrightarrows_{\mathrm{rb}}$ (see Theorem 8.2.9 (Relation between bisimilarity notions)), it follows immediately that $p \leftrightarrows_{\mathrm{rb}} q$ for any closed instantiation $p = q$ of any axiom $s = t$ of $\mathrm{BSP}_\tau(A)$ that is also an axiom of $\mathrm{BSP}(A)$. Hence, $\mathbb{P}(\mathrm{BSP}_\tau(A))_{/\leftrightarrows_{\mathrm{rb}}} \models s = t$ for all these axioms.
For the soundness of Axiom B, it has to be shown that $a.(\tau.(p + q) + p) \leftrightarrows_{\mathrm{rb}} a.(p+q)$ for any action $a \in A_\tau$ and closed terms $p, q \in \mathcal{C}(\mathrm{BSP}_\tau(A))$. Let $R = \{(a.(\tau.(p+q)+p), a.(p+q)), (\tau.(p+q)+p, p+q), (p, p) \mid a \in A_\tau, p, q \in \mathcal{C}(\mathrm{BSP}_\tau(A))\}$. It must be shown that all elements of R satisfy the transfer conditions for branching bisimulation relations of Definition 8.2.3 (Branching bisimilarity) and that the pair $(a.(\tau.(p+q)+p), a.(p+q))$ in addition satisfies the root conditions of Definition 8.2.6 (Rooted branching bisimilarity). For all pairs (p, p) the transfer conditions are satisfied trivially. Next, consider the pairs in R of the form $(\tau.(p + q) + p, p + q)$.

(i) Suppose that $\tau.(p + q) + p \xrightarrow{a} p'$ for some $a \in A_\tau$ and $p' \in \mathcal{C}(\mathrm{BSP}_\tau(A))$. By inspection of the deduction rules, it easily follows that necessarily (1) $p \xrightarrow{a} p'$ or (2) $a \equiv \tau$ and $p' \equiv p + q$. In

https://doi.org/10.1017/CBO9781139195003.012 Published online by Cambridge University Press

the first case, then also $p + q \twoheadrightarrow p + q$ and $p + q \xrightarrow{(a)} p'$, and $(\tau.(p + q) + p, p + q) \in R$ and $(p', p') \in R$. Visually,

$$\begin{array}{ccccc} \tau.(p+q)+p & \xrightarrow{a} & p' & & \\ \vdots & \searrow & & \searrow & \\ p+q & \twoheadrightarrow & p+q & \xrightarrow{(a)} & p' \end{array}$$

In the second case, obviously, $p + q \twoheadrightarrow p + q$, $p + q \xrightarrow{(\tau)} p + q$, $(\tau.(p + q) + p, p + q) \in R$ and $(p + q, p + q) \in R$. Visually,

$$\begin{array}{ccccc} \tau.(p+q)+p & \xrightarrow{\tau} & p' & & \\ \vdots & \searrow & & \searrow & \\ p+q & \twoheadrightarrow & p+q & \xrightarrow{(\tau)} & p' \end{array}$$

(ii) Suppose that $p + q \xrightarrow{a} q'$ for some $a \in A_\tau$ and $q' \in \mathcal{C}(\mathrm{BSP}_\tau(A))$. From the deduction rules, it follows that $\tau.(p + q) + p \xrightarrow{\tau} p + q$. Therefore, $\tau.(p + q) + p \twoheadrightarrow p + q$. Also, $p + q \xrightarrow{(a)} q'$ follows from $p+q \xrightarrow{a} q'$. This means that this case is proven since $(p+q, p+q) \in R$ and $(q', q') \in R$.

$$\begin{array}{ccccc} p+q & \xrightarrow{a} & q' & & \\ \vdots & \searrow & & \searrow & \\ \tau.(p+q)+p & \twoheadrightarrow & p+q & \xrightarrow{(a)} & q' \end{array}$$

(iii) Suppose that $(\tau.(p + q) + p)\downarrow$. Then, necessarily, $p\downarrow$. Then also, $p + q \twoheadrightarrow p + q$ and $(p + q)\downarrow$. Note that $(\tau.(p + q) + p, p + q) \in R$, which proves this case.

(iv) Suppose that $(p + q)\downarrow$. By the deduction rules, $\tau.(p + q) + p \xrightarrow{\tau} p + q$. Therefore, $\tau.(p + q) + p \twoheadrightarrow p + q$. Combined with $(p + q)\downarrow$ and $(p + q, p + q) \in R$ this case is also proven.

Finally, consider the pairs of the form $(a.(\tau.(p + q) + p), a.(p + q))$. As both $(a.(\tau.(p + q) + p))\not\downarrow$ and $(a.(p + q))\not\downarrow$, obviously the root conditions regarding termination are satisfied. The root conditions for transitions are satisfied as the only possible transitions are $a.(\tau.(p + q) + p) \xrightarrow{a} \tau.(p + q) + p$ and $a.(p + q) \xrightarrow{a} p + q$ and $(\tau.(p + q) + p, p + q) \in R$. Visually,

$$\begin{array}{ccc} a.(\tau.(p+q)+p) & \xrightarrow{a} & \tau.(p+q)+p \\ \vdots & & \vdots \\ a.(p+q) & \xrightarrow{a} & p+q \end{array}$$

□

Theorem 8.4.5 (Ground-completeness) Theory $\mathrm{BSP}_\tau(A)$ is a ground-complete axiomatization of the term model $\mathbb{P}(\mathrm{BSP}_\tau(A))_{/\leftrightarrows_{\mathrm{rb}}}$, i.e., for any closed

https://doi.org/10.1017/CBO9781139195003.012 Published online by Cambridge University Press

$\mathrm{BSP}_\tau(A)$-terms p and q, $\mathbb{P}(\mathrm{BSP}_\tau(A))_{/\leftrightarrows_{\mathrm{rb}}} \models p = q$ implies that $\mathrm{BSP}_\tau(A) \vdash p = q$.

The proof of this ground-completeness result needs an alternative characterization of the notion of rooted branching bisimilarity, based on colorings, in line with the characterization of strong bisimilarity given in Exercise 3.1.9.

Definition 8.4.6 (Coloring) A *coloring* of a transition-system space is a mapping from the states of the space to a set of colors. A coloring is *canonical* if two states have the same color if and only if they can be related by a rooted branching bisimulation. A transition system is in *transition-system normal form* if and only if there exists a canonical coloring such that each state of the transition system has a different color and the transition system has no τ-loop $s \xrightarrow{\tau} s$ for any of its states s.

The following property is useful for the proof of Theorem 8.4.5. Two transition systems are isomorphic if and only if there is a one-to-one mapping between their states that preserves transitions and termination options. Let $\cong$ denote isomorphism of transition systems. Recall that transition systems are named by their initial states.

Lemma 8.4.7 For transition systems g and h that are in transition-system normal form, $g \leftrightarrows_{\mathrm{rb}} h$ if and only if $g \cong h$.

Proof The implication from right to left is trivial as the bijection that witnesses the isomorphism is also a rooted branching bisimulation relation. In the other direction, the rooted branching bisimulation relation that proves that g and h are rooted branching bisimilar, is a bijection. It is surjective since any branching bisimulation between two transition systems relates every state of any of the two transition systems to some state in the other transition system. It is injective since every state is related with at most one other state. If two different states in g (h) are related to the same state in h (g), then these two states are rooted branching bisimilar. But then these states have the same color in a canonical coloring, which contradicts the assumption that g (h) is in transition-system normal form. □

Recall the notions of regular transition systems (Definition 3.1.15), regular processes (Definition 5.8.1), and bounded-depth processes (Section 4.5). In line with the reasoning in Sections 4.5, 4.6, and 5.8, it can be shown that closed $\mathrm{BSP}_\tau(A)$-terms can be used to precisely specify all bounded-depth regular processes. As a consequence, it is sufficient to consider only regular transition

systems without any cycles in the remainder of this section. Under this assumption, it is possible to define a rewriting system on transition systems that has the following properties:

(i) normal forms with respect to the rewriting system are transition-system normal forms;
(ii) every rewrite step preserves rooted branching bisimilarity;
(iii) every rewrite step corresponds to a proof in the equational theory $\mathrm{BSP}_\tau(A)$.

Definition 8.4.8 (Double states, manifestly inert transitions) Two states s and t in a transition system are called *double states* if and only if $s \neq t$, $s\!\downarrow$ if and only if $t\!\downarrow$, and for all states u and labels a, $s \xrightarrow{a} u$ if and only if $t \xrightarrow{a} u$. A transition $s \xrightarrow{\tau} t$ is *manifestly inert* if and only if s is not the root of the transition system, if $s\!\downarrow$, then $t\!\downarrow$, and for all states u and labels a such that $a \neq \tau$ or $u \neq t$, if $s \xrightarrow{a} u$, then $t \xrightarrow{a} u$.

Definition 8.4.9 (Rewriting system) The rewriting system consists of the following one-step reductions. Let g and h be transition systems.

(i) Contracting a pair of double states s and t in g: Transition system h is obtained from g by replacing all transitions that lead to s by transitions that lead to t and removing s and all its outgoing transitions from the transition system. Since g and h are both embedded in the underlying transition-system space, this means that g and h both have their own set of states but that there is a bijection between the states of g excluding s and the states of h that preserves transitions and termination options when considering states s and t in g as identical.
(ii) Contracting a manifestly inert transition $s \xrightarrow{\tau} t$ in g: Transition system h is obtained from g by replacing all transitions that lead to s by transitions that lead to t and removing s and all its outgoing transitions from the transition system. As in the previous case, this implies a bijection between the states of g excluding s and the states of h preserving transitions and termination options when considering s and t as identical.

In line with Definitions 2.4.3 (One-step reduction) and 2.4.4 (Reduction relation), a rewrite step is denoted with $\mapsto$ and sequences of rewrites are captured by the reduction relation denoted $\mapsto\!\!\!\!\mapsto$.

Definition 8.4.10 (Term of a transition system) A function $\langle \_ \rangle$ is defined that associates with a transition system a closed $\mathrm{BSP}_\tau(A)$-term as follows. Let r be (the root state of) a transition system.

https://doi.org/10.1017/CBO9781139195003.012 Published online by Cambridge University Press

$$\langle r \rangle = \begin{cases} \sum\limits_{r \xrightarrow{a} s} a.\langle s \rangle & \text{if } r \not\downarrow, \\ \sum\limits_{r \xrightarrow{a} s} a.\langle s \rangle + 1 & \text{if } r \downarrow. \end{cases}$$

Note that this definition is only well-defined for acyclic, regular transition systems. From the definition, it follows immediately that the closed terms associated to isomorphic transition systems are identical and hence derivably equal in $\mathrm{BSP}_\tau(A)$.

Corollary 8.4.11 For transition systems g and h, if $g \cong h$, then $\mathrm{BSP}_\tau(A) \vdash \langle g \rangle = \langle h \rangle$.

The rewriting system on transition systems of Definition 8.4.9 has the following properties:

(i) Normal forms of the rewriting system are transition-system normal forms.

It can be shown that any acyclic regular transition system with branching bisimilar states has at least one pair of states that either form a pair of double states or a manifestly inert τ-transition. Any two bisimilar states closest to (successful or unsuccessful) termination are a pair of double states or a manifestly inert transition. Hence, such a transition system is not a normal form of the rewriting system. Furthermore, any acyclic transition system remains acyclic during rewriting. Therefore, any normal form of the rewriting system has no branching bisimilar states and no τ-loops, which implies that it is in transition-system normal form.

(ii) Every rewrite step preserves rooted branching bisimilarity. That is, for any transition systems g and h,

$$\text{if } g \mapsto h, \text{ then } g \leftrightarrows_{\mathrm{rb}} h. \tag{8.4.1}$$

From $g \mapsto h$, it follows that there exist states s and t such that s and t are double states or such that there is a manifestly inert transition $s \xrightarrow{\tau} t$ and h is the result of replacing all incoming transitions of s by incoming transitions of t and the subsequent removal of s and all associated transitions from g. Let R be the bijection between the states of g minus s and the states of h mentioned in Definition 8.4.9 (Rewriting system). Then, the relation $\{(s, R(t))\} \cup R$ is obviously a rooted branching bisimulation relation between g and h. Hence, $g \leftrightarrows_{\mathrm{rb}} h$.

https://doi.org/10.1017/CBO9781139195003.012 Published online by Cambridge University Press

(iii) Each transition system has a unique normal form (up to isomorphism) with respect to the rewriting system.

Suppose that a transition system g has two normal forms with respect to the rewriting system, say h and h'. Then, by item (i) above this means that h and h' are transition-system normal forms. From property (8.4.1), it follows that $g \leftrightarrows_{\mathrm{rb}} h$ and $g \leftrightarrows_{\mathrm{rb}} h'$. Therefore, also $h \leftrightarrows_{\mathrm{rb}} h'$. Thus, by Lemma 8.4.7, $h \cong h'$.

Now, the following property can be proven.

Lemma 8.4.12 Every rewrite step corresponds to a proof in the theory. For any transition systems g and h,

$$\text{if } g \mapsto h, \text{ then } \mathrm{BSP}_\tau(A) \vdash \langle g \rangle = \langle h \rangle.$$

Proof The rewriting of g into h is due to either the contraction of double states or the contraction of a manifestly inert transition. Suppose that the states involved are s and t. Transition system h is obtained from g by moving all incoming transitions of s to t and by removing s. Let R be the bijection between the states of g minus s and the states of h mentioned in Definition 8.4.9 (Rewriting system). It can be shown that $\mathrm{BSP}_\tau(A) \vdash a.\langle s \rangle = a.\langle t \rangle = a.\langle R(t) \rangle$ for any $a \in A_\tau$. This is trivial for the case that s and t are a pair of double states since the outgoing transitions of s and t have the same label and target states, also $R(t)$ has the same outgoing transitions up to isomorphism of states, and s, t, and $R(t)$ have the same termination options. For the case that $s \xrightarrow{\tau} t$ is a manifestly inert transition, all outgoing transitions of s are also outgoing transitions of t. Then, $\mathrm{BSP}_\tau(A) \vdash a.\langle s \rangle = a.(\tau.\langle t \rangle + p)$ and $\mathrm{BSP}_\tau(A) \vdash \langle t \rangle = p + q$, where p is the result of applying $\langle\rangle$ to s without this manifestly inert transition and where q represents the outgoing transitions of t. Then, $\mathrm{BSP}_\tau(A) \vdash a.\langle s \rangle = a.(\tau.\langle t \rangle + p) = a.(\tau.(p+q)+p) = a.(p+q) = a.\langle t \rangle$. Since $R(t)$ is isomorphic to t, also in this case, it follows by Corollary 8.4.11 that $\mathrm{BSP}_\tau(A) \vdash a.\langle s \rangle = a.\langle t \rangle = a.\langle R(t) \rangle$ for any $a \in A_\tau$.

For states v that are not in between the root state g and state s, obviously $\langle v \rangle \equiv \langle R(v) \rangle$, i.e., the terms are syntactically equal (and hence derivably equal). For any other state v (excluding s), by induction on the distance from state s, it can be proven that $\mathrm{BSP}_\tau(A) \vdash \langle v \rangle = \langle R(v) \rangle$.

- The distance between v and s is one. Assume, $[w\!\downarrow]$ denotes 1 if w is a terminating state, and 0, otherwise. Then,

https://doi.org/10.1017/CBO9781139195003.012 Published online by Cambridge University Press

$$
\begin{aligned}
&\mathrm{BSP}_\tau(A) \vdash \\
&\quad \langle v \rangle = \sum_{v \xrightarrow{a} s} a.\langle s \rangle + \sum_{v \xrightarrow{a} v', v' \not\leftrightarrow s} a.\langle v' \rangle + [v\downarrow] \\
&\qquad = \sum_{v \xrightarrow{a} s} a.\langle t \rangle + \sum_{v \xrightarrow{a} v', v' \not\leftrightarrow s} a.\langle v' \rangle + [v\downarrow] \\
&\qquad = \sum_{R(v) \xrightarrow{a} R(t)} a.\langle R(t) \rangle + \\
&\qquad\qquad \sum_{R(v) \xrightarrow{a} R(v'), v' \not\leftrightarrow t} a.\langle R(v') \rangle + [R(v)\downarrow] \\
&\qquad = \langle R(v) \rangle.
\end{aligned}
$$

- The distance between v and s is more than one. By induction, for all states w such that $v \xrightarrow{a} w$ for some a, $\mathrm{BSP}_\tau(A) \vdash \langle w \rangle = \langle R(w) \rangle$. Then,

$$
\begin{aligned}
\mathrm{BSP}_\tau(A) \vdash \langle v \rangle &= \sum_{v \xrightarrow{a} w} a.\langle w \rangle + [v\downarrow] \\
&= \sum_{v \xrightarrow{a} w} a.\langle R(w) \rangle + [v\downarrow] \\
&= \sum_{R(v) \xrightarrow{a} R(w)} a.\langle R(w) \rangle + [R(v)\downarrow] \\
&= \langle R(v) \rangle.
\end{aligned}
$$

This last result implies that also $\mathrm{BSP}_\tau(A) \vdash \langle g \rangle = \langle R(g) \rangle = \langle h \rangle$, completing the proof. □

So far in this book, closed terms have been used both to refer to syntactical objects in an equational theory and to the transition systems obtained from them in an operational semantics. In the remainder of this section, it is desirable to distinguish these two meanings of a closed term. Therefore, let $[\![p]\!]$ denote the transition system obtained from closed $\mathrm{BSP}_\tau(A)$-term p via the term deduction system of $\mathrm{BSP}_\tau(A)$ defined earlier in this section.

Lemma 8.4.13 For closed $\mathrm{BSP}_\tau(A)$-term p, $\mathrm{BSP}_\tau(A) \vdash \langle [\![p]\!] \rangle = p$.

Proof Via induction on the structure of term p; see Exercise 8.4.4. □

Finally, it is possible to establish the desired ground-completeness result.

Proof (of Theorem 8.4.5 (Ground-completeness)).
Let p and q be closed $\mathrm{BSP}_\tau(A)$-terms. Assume that $\mathbb{P}(\mathrm{BSP}_\tau(A))_{/\leftrightarrow_{\mathrm{rb}}} \models p = q$. Then, by definition, $[\![p]\!] \leftrightarrow_{\mathrm{rb}} [\![q]\!]$. Let g and h be the unique normal forms with respect to the rewriting system of Definition 8.4.9 (Rewriting system) of the transition systems $[\![p]\!]$ and $[\![q]\!]$, respectively:

https://doi.org/10.1017/CBO9781139195003.012 Published online by Cambridge University Press

$$[\![p]\!] \rightarrowtail g \qquad \text{and} \qquad [\![q]\!] \rightarrowtail h.$$

Then, as a direct consequence of Property (8.4.1), $[\![p]\!] \leftrightarrows_{\mathrm{rb}} g$ and $[\![q]\!] \leftrightarrows_{\mathrm{rb}} h$. Since $[\![p]\!] \leftrightarrows_{\mathrm{rb}} [\![q]\!]$, the properties of $\leftrightarrows_{\mathrm{rb}}$ imply that $g \leftrightarrows_{\mathrm{rb}} h$.
Since g and h are in normal form with respect to the rewriting system and since they are also rooted branching bisimilar, it follows (by Lemma 8.4.7) that g and h are isomorphic: $g \cong h$. Therefore, by Corollary 8.4.11, $\mathrm{BSP}_\tau(A) \vdash \langle g \rangle = \langle h \rangle$.
Using Lemma 8.4.12, from $[\![p]\!] \rightarrowtail g$ and $[\![q]\!] \rightarrowtail h$, it follows that $\mathrm{BSP}_\tau(A) \vdash \langle [\![p]\!] \rangle = \langle g \rangle$ and $\mathrm{BSP}_\tau(A) \vdash \langle [\![q]\!] \rangle = \langle h \rangle$. Then,

$$\mathrm{BSP}_\tau(A) \vdash p = \langle [\![p]\!] \rangle = \langle g \rangle = \langle h \rangle = \langle [\![q]\!] \rangle = q,$$

where the first and last equalities are due to Lemma 8.4.13. □

Exercises

8.4.1 Draw the transition systems associated with the following pairs of process terms and construct a rooted branching bisimulation between each of the pairs of transition systems.

(a) $a.(\tau.b.1 + b.1)$ and $a.b.1$;
(b) $a.(\tau.(b.1 + c.1) + b.1)$ and $a.(b.1 + c.1)$;
(c) $a.\tau.(\tau.b.1 + \tau.\tau.b.1)$ and $a.b.1$.

8.4.2 Consider rooted weak bisimilarity as introduced in Exercise 8.2.5. Prove that rooted weak bisimilarity is a congruence on the term algebra $\mathbb{P}(\mathrm{BSP}_\tau(A))$.

8.4.3 Building upon the previous exercise, prove the following, for all closed $\mathrm{BSP}_\tau(A)$-terms p, q:

(a) $\tau.p + p$ is rooted weakly bisimilar to $\tau.p$;
(b) $a.(\tau.p + q)$ is rooted weakly bisimilar to $a.p + a.(\tau.p + q)$.

Are these pairs of terms also rooted branching bisimilar? Motivate your answer.

8.4.4 Prove Lemma 8.4.13.

8.5 Some extensions of $\mathrm{BSP}_\tau(A)$

This section presents extensions of $\mathrm{BSP}_\tau(A)$ with useful operators for abstraction, encapsulation, and projection. Moreover, the choice operators of the process algebra CSP are considered. The treatment is concise, as such extensions follow the same lines as before. However, to prove the various results for

https://doi.org/10.1017/CBO9781139195003.012 Published online by Cambridge University Press

these extended theories, the results of Section 3.2 cannot be used, because these assume strong bisimilarity as the underlying semantic equivalence. Therefore, in the next subsection, all the relevant propositions and theorems are listed, and proofs are asked in the form of exercises. The interested reader is advised to consult the proofs given in Chapter 4 that do not use the framework of Section 3.2. The proofs expected for the results in this section go along the same lines. In Sections 8.5.2, 8.5.3, and 8.5.4, the main results are summarized but for reasons of brevity not explicitly listed.

8.5.1 Abstraction

Recall the calculations for a two-place buffer from Section 7.6. The derivations show that a two-place buffer can be seen as a composition of two one-place buffers, as illustrated in Figure 7.3. The final step of the derivations used a skip operator to skip the execution of internal actions. However, as already explained in the introduction to this chapter, it is more elegant to rename internal actions into unobservable silent actions. Such a renaming, in contrast to the skipping of internal actions, preserves moments of choice. This subsection introduces such a renaming into silent actions, called *abstraction* or *hiding*.

The process theory $(\mathrm{BSP}_\tau + \mathrm{ABS})(A)$ is the extension of the process theory $\mathrm{BSP}_\tau(A)$ with a family of abstraction or hiding operators τ_I for each $I \subseteq A$. The set I contains the atomic actions that need to be abstracted from. The axioms in Table 8.2 closely reflect the operational intuition that all occurrences of prefix operators $a._-$ for which the atomic action is to be hidden ($a \in I$) are replaced by τ-prefix operators.

$(\mathrm{BSP}_\tau + \mathrm{ABS})(A)$
$\mathrm{BSP}_\tau(A)$;
unary: $(\tau_I)_{I \subseteq A}$;
x, y;

$\tau_I(1) = 1$		TI1
$\tau_I(0) = 0$		TI2
$\tau_I(a.x) = a.\tau_I(x)$	if $a \notin I$	TI3
$\tau_I(a.x) = \tau.\tau_I(x)$	if $a \in I$	TI4
$\tau_I(x + y) = \tau_I(x) + \tau_I(y)$		TI5

Table 8.2. Axioms for abstraction ($a \in A_\tau$, $I \subseteq A$).

From the observation that the axioms in Table 8.2 closely follow the structure of $\mathrm{BSP}_\tau(A)$ terms, it follows easily that all occurrences of the abstraction operators can be eliminated.

Theorem 8.5.1 (Elimination) For every $(\mathrm{BSP}_\tau + \mathrm{ABS})(A)$-term p, there is a closed $\mathrm{BSP}_\tau(A)$-term q such that $(\mathrm{BSP}_\tau + \mathrm{ABS})(A) \vdash p = q$.

Proof Exercise 8.5.5. □

The theory also allows the expected conservativity result.

Theorem 8.5.2 (Conservative ground-extension) Theory $(\mathrm{BSP}_\tau + \mathrm{ABS})(A)$ is a conservative ground-extension of process theory $\mathrm{BSP}_\tau(A)$.

Proof Exercise 8.5.6. □

The term model is based on the term algebra for theory $(\mathrm{BSP}_\tau + \mathrm{ABS})(A)$, $\mathbb{P}((\mathrm{BSP}_\tau + \mathrm{ABS})(A)) = (\mathcal{C}((\mathrm{BSP}_\tau + \mathrm{ABS})(A)), +, (a._-)_{a\in A}, \tau._-, (\tau_I)_{I\subseteq A}, 0, 1)$. The term deduction system for $(\mathrm{BSP}_\tau + \mathrm{ABS})(A)$ is obtained by extending the term deduction system for $\mathrm{BSP}_\tau(A)$ with deduction rules for the abstraction operators as defined in Table 8.3. The first deduction rule means that applying abstraction to a process does not alter its termination behavior. The second deduction rule simply states that all behavior that is not abstracted from is still present. The third deduction rule states that all action names that occur in the set I are made unobservable by replacing them by the silent step.

$TDS((\mathrm{BSP}_\tau + \mathrm{ABS})(A))$
$TDS(\mathrm{BSP}_\tau(A))$;

unary: $(\tau_I)_{I\subseteq A}$;

x, x';

$$\frac{x\downarrow}{\tau_I(x)\downarrow} \qquad \frac{x \xrightarrow{a} x' \quad a \notin I}{\tau_I(x) \xrightarrow{a} \tau_I(x')} \qquad \frac{x \xrightarrow{a} x' \quad a \in I}{\tau_I(x) \xrightarrow{\tau} \tau_I(x')}$$

Table 8.3. Term deduction system for $(\mathrm{BSP}_\tau + \mathrm{ABS})(A)$ (with $a \in A_\tau$, $I \subseteq A$).

Proposition 8.5.3 (Congruence) Rooted branching bisimilarity is a congruence on $\mathbb{P}((\mathrm{BSP}_\tau + \mathrm{ABS})(A))$.

Proof Exercise 8.5.7. □

Definition 8.5.4 (Term model of $(\mathrm{BSP}_\tau + \mathrm{ABS})(A)$**)** The term model of theory $(\mathrm{BSP}_\tau + \mathrm{ABS})(A)$ is the quotient algebra $\mathbb{P}((\mathrm{BSP}_\tau + \mathrm{ABS})(A))_{/\leftrightarroweq_{\mathrm{rb}}}$.

https://doi.org/10.1017/CBO9781139195003.012 Published online by Cambridge University Press

Theorem 8.5.5 (Soundness) Theory $(\mathrm{BSP}_\tau + \mathrm{ABS})(A)$ is a sound axiomatization of $\mathbb{P}((\mathrm{BSP}_\tau + \mathrm{ABS})(A))_{/\leftrightarrows_{\mathrm{rb}}}$, i.e., $\mathbb{P}((\mathrm{BSP}_\tau + \mathrm{ABS})(A))_{/\leftrightarrows_{\mathrm{rb}}} \models (\mathrm{BSP}_\tau + \mathrm{ABS})(A)$.

Proof Exercise 8.5.8. □

Theorem 8.5.6 (Ground-completeness) Theory $(\mathrm{BSP}_\tau + \mathrm{ABS})(A)$ is a ground-complete axiomatization of the term model $\mathbb{P}((\mathrm{BSP}_\tau + \mathrm{ABS})(A))_{/\leftrightarrows_{\mathrm{rb}}}$, i.e., for any closed $(\mathrm{BSP}_\tau + \mathrm{ABS})(A)$-terms p and q, $\mathbb{P}((\mathrm{BSP}_\tau + \mathrm{ABS})(A))_{/\leftrightarrows_{\mathrm{rb}}} \models p = q$ implies $(\mathrm{BSP}_\tau + \mathrm{ABS})(A) \vdash p = q$.

Proof Exercise 8.5.9. □

8.5.2 Encapsulation

In Chapter 6, encapsulation operators have been introduced in order to block certain actions from being executed. In Chapter 7, this operator has been used to force certain actions to communicate. In this section, the extension of the process theory $\mathrm{BSP}_\tau(A)$ with such encapsulation operators is discussed. The only relevant difference between the process theories $\mathrm{BSP}(A)$ and $\mathrm{BSP}_\tau(A)$ is the τ-prefix operator. Before, it was assumed that $H \subseteq A$. Now, it should be decided whether it is also allowed to block internal actions, i.e., whether H should be allowed to include τ.

Example 8.5.7 (Encapsulation and silent steps) Assume that it is allowed to block internal actions. Consider the process term $\partial_{\{\tau\}}(a.\tau.1)$. If it is allowed to block silent actions, then the following equality can be established by using the axioms for encapsulation from Table 6.8:

$$\partial_{\{\tau\}}(a.\tau.1) = a.0.$$

However, it is also possible to derive

$$\partial_{\{\tau\}}(a.\tau.1) = \partial_{\{\tau\}}(a.1) = a.1,$$

by applying (a derivative of) the Branching Axiom of Table 8.1 first. Thus, allowing the encapsulation of internal steps leads to the identity

$$a.0 = a.1.$$

Consequently, allowing encapsulation of internal actions results in a process theory that equates successful and unsuccessful termination, and is therefore not a conservative (ground-)extension of $\mathrm{BSP}_\tau(A)$.

https://doi.org/10.1017/CBO9781139195003.012 Published online by Cambridge University Press

As indicated by the example, it is not desirable to allow the encapsulation of internal steps. Therefore, the set of action names to be blocked must still be a subset of A: $H \subseteq A$. Theory $(\mathrm{BSP}_\tau + \mathrm{DH})(A)$ is precisely the union of $(\mathrm{BSP} + \mathrm{DH})(A)$ from Table 6.8 (but assuming $a \in A_\tau$) and theory $\mathrm{BSP}_\tau(A)$ from Table 8.1.

Encapsulation operators can be eliminated from closed $(\mathrm{BSP}_\tau + \mathrm{DH})(A)$-terms to yield a derivably equal closed $\mathrm{BSP}_\tau(A)$-term. Process theory $(\mathrm{BSP}_\tau + \mathrm{DH})(A)$ is furthermore a conservative ground-extension of $\mathrm{BSP}_\tau(A)$. The term deduction system for $(\mathrm{BSP}_\tau + \mathrm{DH})(A)$ is the union of the term deduction systems for $(\mathrm{BSP} + \mathrm{DH})(A)$ (Table 6.10) and $\mathrm{BSP}_\tau(A)$, with $a \in A_\tau$. Theory $(\mathrm{BSP}_\tau + \mathrm{DH})(A)$ is a sound and ground-complete axiomatization of the resulting term model $\mathbb{P}((\mathrm{BSP}_\tau + \mathrm{DH})(A))_{/\leftrightarrows_{\mathrm{rb}}}$.

8.5.3 Projection

Process theory $(\mathrm{BSP}_\tau + \mathrm{PR})(A)$ is the extension of $\mathrm{BSP}_\tau(A)$ with the family of projection operators π_n (for each natural number n). In order to obtain $(\mathrm{BSP}_\tau + \mathrm{PR})(A)$ as a conservative ground-extension of $\mathrm{BSP}_\tau(A)$, it is not possible to simply adopt the axioms of $(\mathrm{BSP} + \mathrm{PR})(A)$ of Table 4.5 with $a \in A_\tau$. This can be illustrated as follows. Consider the following derivations, applying the axioms of $(\mathrm{BSP} + \mathrm{PR})(A)$ on $\mathrm{BSP}_\tau(A)$-terms:

$$\pi_1(a.\tau.1) = a.\pi_0(\tau.1) = a.0$$

and

$$\pi_1(a.\tau.1) = \pi_1(a.1) = a.\pi_0(1) = a.1.$$

Hence, as in Example 8.5.7 (Encapsulation and silent steps), it can be derived that $a.0 = a.1$. As this is an identity between closed $\mathrm{BSP}_\tau(A)$-terms that is not derivable in $\mathrm{BSP}_\tau(A)$, $(\mathrm{BSP}_\tau + \mathrm{PR})(A)$ cannot have the axiom $\pi_0(\tau.x) = 0$. For similar reasons, the axiom $\pi_{n+1}(\tau.x) = \tau.\pi_n(x)$ is not acceptable.

The axioms of $(\mathrm{BSP}_\tau + \mathrm{PR})(A)$ are the axioms of $\mathrm{BSP}_\tau(A)$ with $a \in A_\tau$, the axioms of $(\mathrm{BSP} + \mathrm{PR})(A)$ with $a \in A$, and in addition the axiom presented in Table 8.4.

Theory $(\mathrm{BSP}_\tau + \mathrm{PR})(A)$ allows an elimination result, stating that projection operators can be eliminated from closed $(\mathrm{BSP}_\tau + \mathrm{PR})(A)$-terms. Theory $(\mathrm{BSP}_\tau + \mathrm{PR})(A)$ is also a conservative ground-extension of $\mathrm{BSP}_\tau(A)$.

The term deduction system for $(\mathrm{BSP}_\tau + \mathrm{PR})(A)$ consists of the deduction rules of the term deduction system for $\mathrm{BSP}_\tau(A)$ with $a \in A_\tau$, the deduction rules for $(\mathrm{BSP} + \mathrm{PR})(A)$ with $a \in A$ (Table 4.6), and in addition the

$$\frac{\_\_\,(\mathrm{BSP}_\tau + \mathrm{PR})(A)\,\_\_\_\_\_}{\mathrm{BSP}_\tau(A),\,(\mathrm{BSP} + \mathrm{PR})(A);}$$

$$\frac{\text{-}}{x;}$$

$$\pi_n(\tau.x) = \tau.\pi_n(x) \qquad \text{PR6}$$

Table 8.4. $\mathrm{BSP}_\tau(A)$ with projection ($n \in \mathbf{N}$).

deduction rule given in Table 8.5. Theory $(\mathrm{BSP}_\tau + \mathrm{PR})(A)$ is a sound and ground-complete axiomatization of the term model $\mathbb{P}((\mathrm{BSP}_\tau + \mathrm{PR})(A))_{/\leftrightarrows_{\mathrm{rb}}}$.

$$\frac{\_\_\,TDS((\mathrm{BSP}_\tau + \mathrm{PR})(A))\,\_\_\_\_\_}{TDS(\mathrm{BSP}_\tau(A)),\,TDS((\mathrm{BSP} + \mathrm{PR})(A));}$$

$$\frac{\text{-}}{x, x';}$$

$$\frac{x \xrightarrow{\tau} x'}{\pi_n(x) \xrightarrow{\tau} \pi_n(x')}$$

Table 8.5. Term deduction system for $(\mathrm{BSP}_\tau + \mathrm{PR})(A)$ (with $n \in \mathbf{N}$).

Note that, in the presence of projection operators, one can consider a projective limit model for $(\mathrm{BSP}_\tau + \mathrm{PR})(A)$, similar to the model for theory $(\mathrm{BSP} + \mathrm{PR})(A)$ presented in Section 5.10. It turns out that the notion of projective sequences and the derived model can be adapted to the current setting with silent steps. However, such a model is not possible in a setting with abstraction operators, because abstraction operators cannot be defined on projective sequences. Exercise 8.5.12 investigates this further.

8.5.4 Non-determinism in CSP

In the CSP framework of (Hoare, 1985), bisimilarity is not the preferred notion of equivalence, but rather failures equivalence, to be considered in Section 12.2. As a consequence, the law $a.x + a.y = a.(\tau.x + \tau.y)$ holds in CSP, which means all non-determinism can be reduced to *silent non-determinism* (non-deterministic choices between silent steps).

CSP has two choice operators: $\sqcap$ denoting non-deterministic or *internal* choice, and $\Box$ denoting *external* choice. The internal-choice operator denotes a non-deterministic choice that cannot be influenced by the environment (other

https://doi.org/10.1017/CBO9781139195003.012 Published online by Cambridge University Press

processes in parallel) and can simply be defined in the present setting by the defining axiom

$$x \sqcap y = \tau.x + \tau.y.$$

The definition of $\square$ in the current context with rooted branching bisimilarity as the semantic equivalence is more subtle. It denotes a choice that can be influenced by the environment. If the arguments of the operator have initial silent non-determinism, then these silent steps can be executed without making the choice, and the choice will be made as soon as a visible action occurs. This intuition leads to the operational rules in the two middle rows of Table 8.6, which defines the term deduction system for theory $(\mathrm{BSP}_\tau \sqcap \square)(A)$, $\mathrm{BSP}_\tau(A)$ extended with CSP choice operators, introduced below.

$$\frac{TDS((\mathrm{BSP}_\tau \sqcap \square)(A))}{TDS(\mathrm{BSP}_\tau(A))}$$

-

$x, y, x', y';$

$$\frac{}{x \sqcap y \xrightarrow{\tau} x} \qquad \frac{}{x \sqcap y \xrightarrow{\tau} y}$$

$$\frac{x\downarrow}{x \square y\downarrow} \qquad \frac{x \xrightarrow{a} x'}{x \square y \xrightarrow{a} x'} \qquad \frac{y \xrightarrow{a} y'}{x \square y \xrightarrow{a} y'}$$

$$\frac{y\downarrow}{x \square y\downarrow} \qquad \frac{x \xrightarrow{\tau} x'}{x \square y \xrightarrow{\tau} x' \square y} \qquad \frac{y \xrightarrow{\tau} y'}{x \square y \xrightarrow{\tau} x \square y'}$$

$$\frac{x\downarrow}{x \sqsubset y\downarrow} \qquad \frac{x \xrightarrow{a} x'}{x \sqsubset y \xrightarrow{a} x'} \qquad \frac{x \xrightarrow{\tau} x'}{x \sqsubset y \xrightarrow{\tau} x' \square y}$$

Table 8.6. Term deduction system for $(\mathrm{BSP}_\tau \sqcap \square)(A)$ (with $a \in A$).

An axiomatization of the external-choice operator is not straightforward: similar to the situation with the parallel-composition operator, an auxiliary operator is needed. Let $\sqsubset$ denote the so-called *left-external-choice operator*. It satisfies the operational rules for the external choice that apply to the left operand of the external choice, but not those that apply to the right operand, as can be seen in the bottom row of Table 8.6.

Table 8.7 shows the theory $(\mathrm{BSP}_\tau \sqcap \square)(A)$. It contains besides the defining axiom for the non-deterministic choice, an axiomatization of the CSP external-choice operator, using the left-external-choice operator.

The term deduction system of Table 8.6 can be used to provide a standard term model for theory $(\mathrm{BSP}_\tau \sqcap \square)(A)$ based on rooted branching bisimilarity. Based on the axiomatization of Table 8.7, commutativity and associativity

https://doi.org/10.1017/CBO9781139195003.012 Published online by Cambridge University Press

$\_\_(\mathrm{BSP}_\tau \sqcap \Box)(A)$ _____

$\mathrm{BSP}_\tau(A)$;

binary: $\_ \sqcap \_$, $\_ \Box \_$, $\_ \sqsubset \_$;

x, y, z;

$x \sqcap y = \tau.x + \tau.y$	$x \Box y = x \sqsubset y + y \sqsubset x$
$0 \sqsubset x = 0$	$x \sqsubset 0 = x$
$1 \sqsubset x = 1$	$(x \sqsubset y) \sqsubset z = x \sqsubset (y \Box z)$
$a.x \sqsubset y = a.x$	
$\tau.x \sqsubset y = \tau.(x \Box y)$	
$(x + y) \sqsubset z = x \sqsubset z + y \sqsubset z$	

Table 8.7. $\mathrm{BSP}_\tau(A)$ extended with CSP choice operators (with $a \in A$).

of the external-choice operator can be easily derived. Also, 0 is an identity element of external choice. However, external choice is not idempotent. The internal-choice operator is commutative, but not associative or idempotent. It does not have an identity element.

The axiomatization of the CSP choice operators in the present setting based on bisimilarity uses silent actions. An interesting observation is that the extra τ-identities that are valid in failures semantics, together with the presence of two choice operators allows the elimination of all τ-occurrences from every expression, see (Bergstra *et al.*, 1987), which means that CSP can do without the silent step altogether.

Exercises

8.5.1 Give a derivably equal closed $\mathrm{BSP}_\tau(A)$-term, eliminating silent steps whenever possible, for each of the following process terms:

(a) $\tau_{\{d\}}(a.(d.(b.d.0 + c.d.0) + b.d.0))$;
(b) $\tau_{\{d\}}(a.(d.(b.0 + c.0) + b.1))$;
(c) $\tau_{\{b,c\}}(a.b.0 + a.c.b.1)$;
(d) $\tau_{\{a\}}(b.a.0 + a.c.b.1)$;
(e) $\tau_{\{d\}}(a.(d.b.c.1 + b.c.0))$.

8.5.2 Prove that $(\mathrm{BSP}_\tau + \mathrm{ABS})(A) \vdash \tau_I(\tau_J(p)) = \tau_{I \cup J}(p)$ for any closed $(\mathrm{BSP}_\tau + \mathrm{ABS})(A)$-term p and any $I, J \subseteq A$.

8.5.3 Prove that $(\mathrm{BSP}_\tau + \mathrm{DH})(A) \vdash \partial_G(\partial_H(p)) = \partial_{G \cup H}(p)$ for any closed $(\mathrm{BSP}_\tau + \mathrm{DH})(A)$-term p and any $G, H \subseteq A$.

8.5.4 Give an example to show that $(\mathrm{BSP}_\tau + \mathrm{ABS} + \mathrm{PR})(A) \nvdash \pi_n(\tau_I(x)) = \tau_I(\pi_n(x))$, for $n \in \mathbf{N}$, $I \subseteq A$, and term x.

https://doi.org/10.1017/CBO9781139195003.012 Published online by Cambridge University Press

8.5.5 Prove Theorem 8.5.1 (Elimination of abstraction operators).

8.5.6 Prove Theorem 8.5.2 (Conservativity of $(\mathrm{BSP}_\tau + \mathrm{ABS})(A)$).

8.5.7 Prove Proposition 8.5.3 (Congruence).

8.5.8 Prove Theorem 8.5.5 (Soundness of $(\mathrm{BSP}_\tau + \mathrm{ABS})(A)$).

8.5.9 Prove Theorem 8.5.6 (Ground-completeness of $(\mathrm{BSP}_\tau + \mathrm{ABS})(A)$).

8.5.10 Develop theory $(\mathrm{BSP}_\tau + \mathrm{DH})(A)$. That is, prove elimination and conservativity results, and give a term model proving soundness and ground-completeness.

8.5.11 Develop theory $(\mathrm{BSP}_\tau + \mathrm{PR})(A)$, proving elimination and conservativity results, as well as soundness and ground-completeness results for the standard term model.

8.5.12 Develop a projective limit model for theory $(\mathrm{BSP}_\tau + \mathrm{PR})(A)$, following the developments in Section 5.10. Investigate also the validity of recursion principles. Argue that it is not possible to define abstraction operators on projective sequences.

8.5.13 Develop theory $(\mathrm{BSP}_\tau \sqcap \Box)(A)$, proving elimination and conservativity results, as well as soundness and ground-completeness results for the standard term model based on rooted branching bisimilarity.

8.5.14 Show that theory $(\mathrm{BSP}_\tau \sqcap \Box)(A)$ is a sound axiomatization of the standard term model based on strong bisimilarity, when τ is treated as a normal action. (Note that any identity valid in a term model based on strong bisimilarity is also valid in a term model based on (rooted) branching bisimilarity.)

8.5.15 Derive the following equations from $(\mathrm{BSP}_\tau \sqcap \Box)(A)$:

(a) $x \sqcap y = y \sqcap x$;
(b) $x \Box y = y \Box x$;
(c) $x \Box (y \Box z) = (x \Box y) \Box z$;
(d) $x \Box 0 = x$.

Provide counterexamples to show that the following equations are not derivable:

(a) $x \sqcap x = x$;
(b) $x \sqcap (y \sqcap z) = (x \sqcap y) \sqcap z$;
(c) $x \Box x = x$;
(d) $x \Box (y \sqcap z) = (x \Box y) \sqcap (x \Box z)$;
(e) $x \sqcap (y \Box z) = (x \sqcap y) \Box (x \sqcap z)$.

https://doi.org/10.1017/CBO9781139195003.012 Published online by Cambridge University Press

8.6 TCP with silent steps

The basic theory $\mathrm{BSP}_\tau(A)$ developed in this chapter can be extended with the various features introduced so far throughout this book. This section introduces process theory $\mathrm{TCP}_\tau(A, \gamma)$, which extends $\mathrm{BSP}_\tau(A)$ with sequential composition, parallel composition, encapsulation, and abstraction, providing a rich process theory, called the Theory of Communicating Processes with silent steps, that includes many useful ingredients for specifying and reasoning about processes. The axioms of theory $\mathrm{TCP}_\tau(A, \gamma)$ are the axioms of process theory $\mathrm{TCP}(A, \gamma)$ of Chapter 7 with $a \in A_\tau$ and $H \subseteq A$, the axioms of theory $(\mathrm{BSP}_\tau + \mathrm{ABS})(A)$ of the previous section, and two additional Axioms of Standard Concurrency SC9 and SC10. For readability, Table 8.8 gives a complete overview of the signature and the axioms of process theory $\mathrm{TCP}_\tau(A, \gamma)$. Communications involving the silent step τ are always assumed to be undefined. This is captured axiomatically by Axiom SC10. (Exercise 8.6.8 illustrates why this assumption is necessary.) Axiom SC9 is a counterpart of the Branching Axiom B for the auxiliary left-merge operator. From Axiom SC9, the equality $x \between (\tau.(y + z) + y) = x \between (y + z)$ can be proven (see Exercise 8.6.4).

The extensions of $\mathrm{TCP}_\tau(A, \gamma)$ with respect to theory $\mathrm{BSP}_\tau(A)$ allow for the convenient specification of communicating processes but do not add to the expressiveness of the theory.

Theorem 8.6.1 (Elimination) For any closed $\mathrm{TCP}_\tau(A, \gamma)$-term p, there is a closed $\mathrm{BSP}_\tau(A)$-term q such that $\mathrm{TCP}_\tau(A, \gamma) \vdash p = q$.

Proof Exercise 8.6.9. □

Theorem 8.6.2 (Conservative ground-extension) Theory $\mathrm{TCP}_\tau(A, \gamma)$ is a conservative ground-extension of theory $\mathrm{BSP}_\tau(A)$.

Proof Exercise 8.6.10. □

Theorem 8.6.3 (Standard concurrency) For closed $\mathrm{TCP}_\tau(A, \gamma)$-terms, the Axioms of Standard Concurrency SC2–SC10 are derivable from the other axioms of $\mathrm{TCP}_\tau(A, \gamma)$.

Proof For Axioms SC2–SC8, the proofs of Theorems 7.4.4 and 7.7.1 (Standard concurrency in $\mathrm{BCP}(A, \gamma)$ and $\mathrm{TCP}(A, \gamma)$) carry over to the current setting. The proof for the two new axioms goes via structural induction. First, the result for SC10 is proven, and then this can be used in the proof of SC9. See Exercise 8.6.3. □

https://doi.org/10.1017/CBO9781139195003.012 Published online by Cambridge University Press

$\mathrm{TCP}_\tau(A,\gamma)$

constant: 0, 1;

unary: $(a.\_)_{a\in A_\tau}$, $(\partial_H)_{H\subseteq A}$, $(\tau_I)_{I\subseteq A}$;

binary: $\_+\_$, $\_\cdot\_$, $\_\parallel\_$, $\_\between\_$, $\_\mid\_$;

x, y, z;

$x+y=y+x$	A1		$x+0=x$	A6
$(x+y)+z=x+(y+z)$	A2		$0\cdot x=0$	A7
$x+x=x$	A3		$x\cdot 1=x$	A8
$(x+y)\cdot z=x\cdot z+y\cdot z$	A4		$1\cdot x=x$	A9
$(x\cdot y)\cdot z=x\cdot(y\cdot z)$	A5		$a.x\cdot y=a.(x\cdot y)$	A10
$a.(\tau.(x+y)+x)=a.(x+y)$	B			
$\partial_H(1)=1$	D1		$\tau_I(1)=1$	TI1
$\partial_H(0)=0$	D2		$\tau_I(0)=0$	TI2
$\partial_H(a.x)=0$	D3	if $a\in H$	$\tau_I(a.x)=a.\tau_I(x)$	TI3 if $a\notin I$
$\partial_H(a.x)=a.\partial_H(x)$	D4	if $a\notin H$	$\tau_I(a.x)=\tau.\tau_I(x)$	TI4 if $a\in I$
$\partial_H(x+y)=\partial_H(x)+\partial_H(y)$	D5		$\tau_I(x+y)=\tau_I(x)+\tau_I(y)$	TI5
$x\parallel y=x\between y+y\between x+x\mid y$	M		$x\mid y=y\mid x$	SC1
			$x\parallel 1=x$	SC2
$0\between x=0$	LM1		$1\mid x+1=1$	SC3
$1\between x=0$	LM2		$(x\parallel y)\parallel z=x\parallel(y\parallel z)$	SC4
$a.x\between y=a.(x\parallel y)$	LM3		$(x\mid y)\mid z=x\mid(y\mid z)$	SC5
$(x+y)\between z=x\between z+y\between z$	LM4		$(x\between y)\between z=x\between(y\between z)$	SC6
$0\mid x=0$	CM1		$(x\mid y)\between z=x\mid(y\between z)$	SC7
$(x+y)\mid z=x\mid z+y\mid z$	CM2		$x\between 0=x\cdot 0$	SC8
$1\mid 1=1$	CM3		$x\between\tau.y=x\between y$	SC9
$a.x\mid 1=0$	CM4		$x\mid\tau.y=0$	SC10
$a.x\mid b.y=c.(x\parallel y)$		if $\gamma(a,b)=c$	CM5	
$a.x\mid b.y=0$		if $\gamma(a,b)$ is not defined	CM6	

Table 8.8. The process theory $\mathrm{TCP}_\tau(A,\gamma)$ (with $a,b,c\in A_\tau$, $H,I\subseteq A$).

As in Section 7.4, an expansion theorem can be proven that allows reasoning about parallel compositions.

Theorem 8.6.4 (Expansion theorem) Let set $I\subset\mathbf{N}$ be a finite, non-empty set of natural numbers, and t_i, for all $i\in I$, arbitrary $\mathrm{TCP}_\tau(A,\gamma)$-terms.

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

Proof The proof is similar to the proof of the expansion theorem in the previous chapter. See Exercise 8.6.11. □

Also in line with the development of theory $\mathrm{BCP}(A,\gamma)$ of the previous

https://doi.org/10.1017/CBO9781139195003.012 Published online by Cambridge University Press

chapter, it is possible to add the Free-Merge and Handshaking Axioms to the theory, resulting in theories $(\mathrm{TCP}_\tau + \mathrm{FMA})(A, \emptyset)$ and $(\mathrm{TCP}_\tau + \mathrm{HA})(A, \gamma)$, respectively. Each of these two theories has its own variant of the expansion theorem.

Theorem 8.6.5 (Expansion theorem for free merge) Let set $I \subset \mathbf{N}$ be some finite, non-empty index set of natural numbers, and t_i, for all $i \in I$, arbitrary $(\mathrm{TCP}_\tau + \mathrm{FMA})(A, \emptyset)$-terms.

$$(\mathrm{TCP}_\tau + \mathrm{FMA})(A, \emptyset) \vdash \|_{i \in I} t_i = \sum_{i \in I} t_i \mathbin{\lfloor\!\lfloor} (\|_{j \in I \setminus \{i\}} t_j) + |_{i \in I} t_i.$$

Proof Exercise 8.6.12. □

Theorem 8.6.6 (Expansion theorem for 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 $(\mathrm{TCP}_\tau + \mathrm{HA})(A, \gamma)$-terms.

$$(\mathrm{TCP}_\tau + \mathrm{HA})(A, \gamma) \vdash \|_{i \in I} t_i = \sum_{i \in I} t_i \mathbin{\lfloor\!\lfloor} (\|_{j \in I \setminus \{i\}} t_j) + \sum_{i,j \in I, i \neq j} (t_i \mid t_j) \mathbin{\lfloor\!\lfloor} (\|_{k \in I \setminus \{i,j\}} t_k) + |_{i \in I} t_i.$$

Proof Exercise 8.6.13. □

The development of the standard term model for $\mathrm{TCP}_\tau(A, \gamma)$ follows the usual pattern. The relevant definitions and results are summarized for the sake of completeness.

The term algebra is the algebra $\mathbb{P}(\mathrm{TCP}_\tau(A, \gamma)) = (\mathcal{C}(\mathrm{TCP}_\tau(A, \gamma)), +, \cdot, \|, \mathbin{\lfloor\!\lfloor}, \mid, (a._-)_{a \in A}, \tau._-, (\partial_H)_{H \subseteq A}, (\tau_I)_{I \subseteq A}, 0, 1)$. The term deduction system for $\mathrm{TCP}_\tau(A, \gamma)$ is the union of the term deduction systems for $\mathrm{TCP}(A, \gamma)$ and $(\mathrm{BSP}_\tau + \mathrm{ABS})(A)$.

Proposition 8.6.7 (Congruence) Rooted branching bisimilarity is a congruence on the algebra $\mathbb{P}(\mathrm{TCP}_\tau(A, \gamma))$.

Proof Exercise 8.6.14. □

Definition 8.6.8 (Term model) The term model of $\mathrm{TCP}_\tau(A, \gamma)$ is the quotient algebra $\mathbb{P}(\mathrm{TCP}_\tau(A, \gamma))_{/\leftrightarrows_{\mathrm{rb}}}$.

Theorem 8.6.9 (Soundness) Theory $\mathrm{TCP}_\tau(A, \gamma)$ is a sound axiomatization of the algebra $\mathbb{P}(\mathrm{TCP}_\tau(A, \gamma))_{/\leftrightarrows_{\mathrm{rb}}}$, i.e., $\mathbb{P}(\mathrm{TCP}_\tau(A, \gamma))_{/\leftrightarrows_{\mathrm{rb}}} \models \mathrm{TCP}_\tau(A, \gamma)$.

https://doi.org/10.1017/CBO9781139195003.012 Published online by Cambridge University Press

Proof Exercise 8.6.15. □

Theorem 8.6.10 (Ground-completeness) Theory $\mathrm{TCP}_\tau(A,\gamma)$ is a ground-complete axiomatization of model $\mathbb{P}(\mathrm{TCP}_\tau(A,\gamma))_{/\leftrightarrows_{\mathrm{rb}}}$, i.e., for any closed $\mathrm{TCP}_\tau(A,\gamma)$-terms p and q, $\mathbb{P}(\mathrm{TCP}_\tau(A,\gamma))_{/\leftrightarrows_{\mathrm{rb}}} \models p = q$ implies $\mathrm{TCP}_\tau(A,\gamma) \vdash p = q$.

Proof Exercise 8.6.16. □

Exercises

8.6.1 Let $A = \{a, b, c\}$ with $\gamma(a, b) = c$ be the only defined communication. Eliminate all occurrences of the communication operators $\parallel$, $\between$, and $|$, and as many τ-prefix operator occurrences as possible from the process term $\tau.a.b.1 \parallel \tau.(\tau.a.1 + \tau.b.1)$.

8.6.2 Prove the following equalities, for $a \in A_\tau$ and $\mathrm{TCP}_\tau(A,\gamma)$-terms x, y:

(a) $\mathrm{TCP}_\tau(A,\gamma) \vdash a.(\tau.1 \parallel x) = a.x$;
(b) $\mathrm{TCP}_\tau(A,\gamma) \vdash a.(\tau.x \parallel y) = a.(x \parallel y)$.

8.6.3 Prove Theorem 8.6.3 (Standard concurrency) for Axioms SC9 and SC10.

8.6.4 Prove the following equalities, for $\mathrm{TCP}_\tau(A,\gamma)$-terms x, y, z:

(a) $\mathrm{TCP}_\tau(A,\gamma) \vdash x \parallel \tau.y = \tau.(x \parallel y) + x \between y$;
(b) $\mathrm{TCP}_\tau(A,\gamma) \vdash x \between (\tau.y + y) = x \between y$;
(c) $\mathrm{TCP}_\tau(A,\gamma) \vdash x \between (\tau.(y + z) + y) = x \between (y + z)$.

8.6.5 Assume that $I \subseteq A$ and that no element of I can communicate, i.e., $\gamma(i, a)$ and $\gamma(a, i)$ are undefined for any $a \in A$ and $i \in I$. Prove that for all closed $\mathrm{TCP}_\tau(A,\gamma)$-terms p and q:

$$\mathrm{TCP}_\tau(A,\gamma) \vdash \tau_I(p \parallel q) = \tau_I(\tau_I(p) \parallel \tau_I(q)).$$

Assume, in addition, that none of the elements in I is the result of a communication, i.e., $\gamma(a, b) \notin I$ for any $a, b \in A$. Prove that for all closed $\mathrm{TCP}_\tau(A,\gamma)$-terms p and q:

$$\mathrm{TCP}_\tau(A,\gamma) \vdash \tau_I(p \parallel q) = \tau_I(p) \parallel \tau_I(q).$$

8.6.6 Prove, for arbitrary $\mathrm{TCP}_\tau(A,\gamma)$-terms x and y, closed $\mathrm{TCP}_\tau(A,\gamma)$-term p, and action $a \in A_\tau$ that $\mathrm{TCP}_\tau(A,\gamma) \vdash a.p \cdot (\tau.(x + y) + x) = a.p \cdot (x + y)$. As a generalization, prove, for arbitrary $\mathrm{TCP}_\tau(A,\gamma)$-terms x and y, that $\mathrm{TCP}_\tau(A,\gamma) \vdash p \cdot (\tau.(x + y) + x) = p \cdot (x + y)$

for any closed $\mathrm{TCP}_\tau(A,\gamma)$-term p of the form $p = \sum_{i<n} a_i.p_i$ with $a_i \in A_\tau$ and p_i a closed $\mathrm{TCP}_\tau(A,\gamma)$-term for some $n \in \mathbf{N}$.

8.6.7 Let x, y, z be $\mathrm{TCP}_\tau(A,\gamma)$-terms. Prove that Axiom B can be derived from $x \between (\tau.(y+z)+y) = x \between (y+z)$ and the other axioms of $\mathrm{TCP}_\tau(A,\gamma)$.

8.6.8 Consider the assumption that communication with τ is always undefined. Show that without this assumption Axiom SC10 is not valid in the standard term model. Consider theory $\mathrm{TCP}_\tau(A,\gamma)$ without Axiom SC10 and without the assumption that communication with τ is impossible. Where does the standard development of this theory go wrong?

8.6.9 Prove Theorem 8.6.1 (Elimination).

8.6.10 Prove Theorem 8.6.2 (Conservative ground-extension).

8.6.11 Prove Theorem 8.6.4 (Expansion theorem).

8.6.12 Prove Theorem 8.6.5 (Expansion theorem for free merge).

8.6.13 Prove Theorem 8.6.6 (Expansion theorem for handshaking communication).

8.6.14 Prove Proposition 8.6.7 (Congruence).

8.6.15 Prove Theorem 8.6.9 (Soundness).

8.6.16 Prove Theorem 8.6.10 (Ground-completeness).

8.6.17 Specialize the set of actions A into a set of names N, a set of co-names $\bar{N}$, and a set of communications C such that for each $n \in N$, there is exactly one $\bar{n} \in \bar{N}$ and exactly one $n_c \in C$. The communication function γ has $\gamma(n,\bar{n}) = \gamma(\bar{n},n) = n_c$ and is not defined otherwise. Now define the CCS parallel composition operator $\between^{\mathrm{CCS}}$ by the following defining axiom, with x and y arbitrary terms:

$$x \between^{\mathrm{CCS}} y = \tau_C(x \between y).$$

Derive the CCS *expansion law* (see (Milner, 1980; Milner, 1989)) from the theory $\mathrm{TCP}_\tau(A,\gamma)$ extended with this defining axiom, and where A and γ are defined as in this exercise.

8.7 Iteration and divergence

In Section 4.6, prefix iteration a^*x for action a and term x has been considered. The introduction of the silent step τ implies an additional action-prefix operator $\tau._-$; so, it is natural to also have a look at τ-prefix iteration τ^*_-, sometimes called *divergence*, capturing the fact that it allows an infinite sequence of τ actions to occur. The theory $\mathrm{BSP}^*_\tau(A)$ is the union of theories $\mathrm{BSP}^*(A)$ of Section 4.6 and $\mathrm{BSP}_\tau(A)$, extended with the τ-prefix iteration; see Table 8.9.

https://doi.org/10.1017/CBO9781139195003.012 Published online by Cambridge University Press

The term deduction system for $\mathrm{BSP}^*_\tau(A)$ is simply the term deduction system of $\mathrm{BSP}^*(A)$ but with $a \in A_\tau$.

$$\frac{\mathrm{BSP}^*_\tau(A)}{\mathrm{BSP}^*(A), \mathrm{BSP}_\tau(A)}$$

unary: $\tau^*\_$;

–

Table 8.9. The process theory $\mathrm{BSP}^*_\tau(A)$.

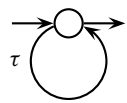


Fig. 8.10. τ^*1.

However, more can be said about τ-prefix iteration. Consider the process τ^*1, depicted in Figure 8.10. This process is branching bisimilar to the process 1 (but not rooted branching bisimilar!). This suggests that τ-loops can be removed. Since the standard term model is based on rooted branching bisimilarity, in the term model, only τ-loops not at the root can be removed entirely; τ-loops at the root can be reduced to a single τ-step. This fact can be captured as the *Fair-Iteration* Axiom; see Table 8.10. It can be shown, see Exercise 8.7.1, that the Fair-Iteration Axiom implies that $(\mathrm{BSP}^*_\tau + \mathrm{FI})(A) \vdash a.p \cdot \tau^* x = a.p \cdot x$ for all $(\mathrm{BSP}^*_\tau + \mathrm{FI})(A)$-terms x and closed $(\mathrm{BSP}^*_\tau + \mathrm{FI})(A)$-terms p. The Fair-Iteration Axiom expresses that τ-prefix iteration behaves fairly, in the sense that no infinite sequence of τ-steps occurs, or in other words that divergence does not occur. However, it needs to be stressed that the Fair-Iteration Axiom happens to hold in the standard term model of $\mathrm{BSP}^*_\tau(A)$ with rooted branching bisimilarity as the underlying equivalence, which means that this model is also a model of $(\mathrm{BSP}^*_\tau + \mathrm{FI})(A)$. Other models of $\mathrm{BSP}^*_\tau(A)$ exist where fair iteration does not hold, which means that these models are not models of $(\mathrm{BSP}^*_\tau + \mathrm{FI})(A)$.

It is interesting to consider a model that does not allow to remove τ-loops. Abstracting from τ-loops implies that processes without divergence are equivalent to processes with divergence. This is not always desirable. A model for BSP^*_τ that does not abstract from divergence can be constructed in the same way as the standard term model, but using a different semantic equivalence at the basis.

https://doi.org/10.1017/CBO9781139195003.012 Published online by Cambridge University Press

$$\frac{(\mathrm{BSP}^*_\tau + \mathrm{FI})(A)}{\mathrm{BSP}^*_\tau(A)}$$

-

x;

$$\tau.\tau^* x = \tau.x \qquad \text{FI}$$

Table 8.10. Fair iteration.

Definition 8.7.1 (Branching bisimilarity with explicit divergence) A binary relation R on the set of states S of a transition-system space is a *branching bisimulation relation with explicit divergence* if and only if R is a branching bisimulation relation on S and in addition the following conditions hold:

(i) for all states $s, t \in S$, whenever $(s, t) \in R$ and $s \xrightarrow{\tau} s_0 \xrightarrow{\tau} s_1 \cdots$ is an infinite τ-path such that $(s_i, t) \in R$ for all $i \in \mathbf{N}$, then there are states $t_0, t_1, \ldots$ such that $t \xrightarrow{\tau} t_0 \xrightarrow{\tau} t_1 \cdots$ and $(s_i, t_j) \in R$ for all $i, j \in \mathbf{N}$;

(ii) vice versa, for all states $s, t \in S$, whenever $(s, t) \in R$ and $t \xrightarrow{\tau} t_0 \xrightarrow{\tau} t_1 \cdots$ is an infinite τ-path such that $(s, t_i) \in R$ for all $i \in \mathbf{N}$, then there are states $s_0, s_1, \ldots$ such that $s \xrightarrow{\tau} s_0 \xrightarrow{\tau} s_1 \cdots$ and $(s_j, t_i) \in R$ for all $i, j \in \mathbf{N}$.

This definition might seem very restrictive, but this is not really the case. As before, it is possible to define a rootedness condition for this notion, so that rooted branching bisimilarity with explicit divergence becomes a congruence relation on the term algebra. As a consequence, a model of $\mathrm{BSP}^*_\tau(A)$ is obtained where the Fair-Iteration Axiom does not hold; divergence cannot be abstracted away.

It is also possible to proceed in another way, by treating divergence as chaos. This is an extreme standpoint that states that, as soon as divergent behavior is possible, nothing more can be said about the process, no observation is possible. This means that any process with divergence is identified with the completely arbitrary process that can only do internal actions, called the *chaos* process. This situation is characterized by the axiom given in Table 8.11.

Proposition 8.7.2 (Chaos) The following equalities are derivable from theory $(\mathrm{BSP}^*_\tau + \mathrm{CH})(A)$, for any $(\mathrm{BSP}^*_\tau + \mathrm{CH})(A)$-terms x and y.

(i) $\tau^* x = x + \tau^* 0$;
(ii) $\tau^* x + y = \tau^* 0$.

https://doi.org/10.1017/CBO9781139195003.012 Published online by Cambridge University Press

$\frac{}{(\mathrm{BSP}^*_\tau + \mathrm{CH})(A)}$
$\frac{}{\mathrm{BSP}^*_\tau(A)}$
$\frac{}{x;}$

$$\frac{}{\tau^* x = \tau^* 0} \quad \mathrm{CH}$$

Table 8.11. Catastrophic divergence.

Proof The derivations are as follows, where the second derivation uses the first identity in the second step. $(\mathrm{BSP}^*_\tau + \mathrm{CH})(A) \vdash$

(i) $\tau^* x = x + \tau.\tau^* x = x + x + \tau.\tau^* x = x + \tau^* x = x + \tau^* 0$;
(ii) $\tau^* x + y = \tau^* 0 + y = \tau^* y = \tau^* 0$. □

This proposition, in particular the second identity, characterizes process $\tau^* x$ as chaos, that makes any determination of the subsequent or alternative process behavior impossible. The standard term model of $\mathrm{BSP}^*_\tau(A)$ does not validate the Chaos Axiom CH, but it is possible to find a model for the theory $\mathrm{BSP}^*_\tau(A)$ that does satisfy this axiom, and which is therefore a model of $(\mathrm{BSP}^*_\tau + \mathrm{CH})(A)$ (see Exercise 8.7.6).

Exercises

8.7.1 Show that the Fair-Iteration Axiom can be used to prove that $(\mathrm{BSP}^*_\tau + \mathrm{FI})(A) \vdash a.p \cdot \tau^* x = a.p \cdot x$ for all $(\mathrm{BSP}^*_\tau + \mathrm{FI})(A)$-terms x and closed $(\mathrm{BSP}^*_\tau + \mathrm{FI})(A)$-terms p. As a generalization, prove that $(\mathrm{BSP}^*_\tau + \mathrm{FI})(A) \vdash p \cdot \tau^* x = p \cdot x$ for all $(\mathrm{BSP}^*_\tau + \mathrm{FI})(A)$-terms x and closed terms p of the form $\sum_{i<n} a_i.p_i$ with $a_i \in A_\tau$ and closed terms p_i for some $n \in \mathbf{N}$.

8.7.2 In a transition system, a *divergent* state is a state from which an infinite series of τ-steps is possible. Find two transition systems that are branching bisimilar, by a branching bisimulation that relates divergent states to divergent states only, but such that the transition systems are not branching bisimilar with explicit divergence.

8.7.3 Verify that rooted branching bisimilarity with explicit divergence is a congruence relation on the term algebra for $\mathrm{BSP}^*_\tau(A)$. Verify that the resulting quotient algebra is a model for $\mathrm{BSP}^*_\tau(A)$, but that this algebra does not validate the Fair-Iteration Axiom FI (and is hence not a model of $(\mathrm{BSP}^*_\tau + \mathrm{FI})(A)$).

8.7.4 A weaker version of the Fair-Iteration Axiom is the equation

https://doi.org/10.1017/CBO9781139195003.012 Published online by Cambridge University Press

$$\tau^*\tau.x = \tau.x$$

that only removes so-called *unstable* divergence (i.e., divergence can only be escaped by means of a process that starts with a silent step). Show that the Fair-Iteration Axiom implies this equation. On the other hand, try to come up with a model for BSP^*_τ that satisfies this equation but not the Fair-Iteration Axiom.

8.7.5 Prove that $(\mathrm{BSP}^*_\tau + \mathrm{CH})(A) \vdash$

(a) $\tau^*(x + y) = \tau^*x + \tau^*y$;
(b) $\tau^*x + y = \tau^*x$.

8.7.6 Construct a model for $(\mathrm{BSP}^*_\tau + \mathrm{CH})(A)$.

8.8 Recursion and fair abstraction

The previous section illustrates that recursive and silent behavior form an interesting combination. This section investigates the extension of the general theory $\mathrm{TCP}_\tau(A, \gamma)$ with general recursion.

A crucial concept in any context with recursion is the concept of guardedness. In Chapter 5, an occurrence of a recursion variable in a term has been defined to be guarded if it occurs in a subterm of the form $a.s$ for some action $a \in A$ and term s. Now that a prefix operator $\tau._-$ has been added, it should be discussed whether this operator can also act as a guard of a recursion variable.

To this end, consider the recursive equation $X = \tau.X$. Considering the term model of $\mathrm{TCP}_\tau(A, \gamma)$ of Section 8.6, all processes $[\tau.p]_{\leftrightarrows_{\mathrm{rb}}}$ are solutions for X, for any closed $\mathrm{TCP}_\tau(A, \gamma)$-term p, since $\tau.p \leftrightarrows_{\mathrm{rb}} \tau.\tau.p$. Hence, the silent step cannot be considered a guard for an occurrence of a recursion variable.

Also, the abstraction operator, as introduced in Section 8.5.1, can cause problems when considering recursion, and guardedness in particular.

Example 8.8.1 (Direct abstraction from a guard) Consider the recursive specification $E = \{X = \tau_I(i.X)\}$ where $i \in I$. With the definition of guardedness as given before, this recursive specification is guarded. Nevertheless, this equation has many different solutions: for all distinct actions a and b not in I, processes $[\tau.a.1]_{\leftrightarrows_{\mathrm{rb}}}$ and $[\tau.b.1]_{\leftrightarrows_{\mathrm{rb}}}$ are solutions for X. The reason for this problem is the occurrence of the abstraction operator, which is problematic because it abstracts from an atomic action that acts as a guard for a recursion variable.

Example 8.8.2 (Indirect abstraction from a guard) Consider the recursive specification $E = \{X = i.\tau_I(X)\}$ where $i \in I$. Although initially, the action i

https://doi.org/10.1017/CBO9781139195003.012 Published online by Cambridge University Press

acts as a guard for the occurrence of X on the right-hand side, all subsequent occurrences of i are abstracted away, as the recursive call to X is in the scope of the abstraction operator. Also this recursive specification has many solutions. For example, for different a and b not in I, the processes $[i.a.1]_{\leftrightarrows_{\mathrm{rb}}}$ and $[i.b.1]_{\leftrightarrows_{\mathrm{rb}}}$ are solutions for X.

These difficulties make it necessary to restrict the notion of guardedness. A recursive specification will only be said to be guarded if it can be brought into a form (by applying the equations and axioms of the theory) where all variables on the right-hand side are guarded by an atomic action different from τ and in which no abstraction operator occurs. The following definition replaces the corresponding part of Definition 5.5.5 (Guardedness, part 1). The other parts of Definitions 5.5.5 and 5.5.8 (Guardedness, parts 1 and 2) carry over to the current context without change.

Definition 8.8.3 (Guardedness in $\mathrm{TCP}_{\tau,\mathrm{rec}}(A,\gamma)$**)** An occurrence of a variable x in a $\mathrm{TCP}_{\tau,\mathrm{rec}}(A,\gamma)$-term s is *guarded* if and only if the abstraction operator does not occur in s and x occurs in a subterm of the form $a.t$ for some action $a \in A$ (so not equal to τ) and $\mathrm{TCP}_{\tau,\mathrm{rec}}(A,\gamma)$-term t.

With this restricted definition of guardedness, the familiar results can be established. The term model of the theory extended with recursion constants is defined as before. Proofs are left to the reader. They go along the lines of similar proofs given earlier in this book and proofs in (Baeten *et al.*, 1987b).

Theorem 8.8.4 (Validity of RDP**)** Recursion principle RDP is not valid in the standard term model but it is valid in the term model for the theory extended with recursion constants: $\mathbb{P}(\mathrm{TCP}_{\tau}(A,\gamma))_{/\leftrightarrows_{\mathrm{rb}}} \not\models \mathrm{RDP}$ and $\mathbb{P}(\mathrm{TCP}_{\tau,\mathrm{rec}}(A,\gamma))_{/\leftrightarrows_{\mathrm{rb}}} \models \mathrm{RDP}$.

Theorem 8.8.5 (Validity of RDP^-**)** The principle RDP^- is not valid in the standard term model but it is valid in the term model with recursion constants: $\mathbb{P}(\mathrm{TCP}_{\tau}(A,\gamma))_{/\leftrightarrows_{\mathrm{rb}}} \not\models \mathrm{RDP}^-$ and $\mathbb{P}(\mathrm{TCP}_{\tau,\mathrm{rec}}(A,\gamma))_{/\leftrightarrows_{\mathrm{rb}}} \models \mathrm{RDP}^-$.

Theorem 8.8.6 (Validity of RSP**)** Recursion principle RSP is valid in both the term models with and without recursion constants: $\mathbb{P}(\mathrm{TCP}_{\tau}(A,\gamma))_{/\leftrightarrows_{\mathrm{rb}}} \models \mathrm{RSP}$ and $\mathbb{P}(\mathrm{TCP}_{\tau,\mathrm{rec}}(A,\gamma))_{/\leftrightarrows_{\mathrm{rb}}} \models \mathrm{RSP}$.

The extension of theory $\mathrm{TCP}_{\tau,\mathrm{rec}}(A,\gamma)$ with projection operators, needed for the formulation of the recursion principles AIP and AIP^-, can be obtained as outlined in Section 8.5.3.

https://doi.org/10.1017/CBO9781139195003.012 Published online by Cambridge University Press

Theorem 8.8.7 (Validity of AIP**)** Recursion principle AIP is valid in the standard term model but it is not valid in the term model with recursion constants: $\mathbb{P}((\mathrm{TCP}_\tau + \mathrm{PR})(A,\gamma))_{/\leftrightarrows_{\mathrm{rb}}} \models \mathrm{AIP}$ and $\mathbb{P}((\mathrm{TCP}_\tau + \mathrm{PR})_{\mathrm{rec}}(A,\gamma))_{/\leftrightarrows_{\mathrm{rb}}} \not\models \mathrm{AIP}$.

Theorem 8.8.8 (Validity of AIP^-**)** Principle AIP^- is valid in both the term models with and without recursion constants: $\mathbb{P}((\mathrm{TCP}_\tau + \mathrm{PR})(A,\gamma))_{/\leftrightarrows_{\mathrm{rb}}} \models \mathrm{AIP}^-$ and $\mathbb{P}((\mathrm{TCP}_\tau + \mathrm{PR})_{\mathrm{rec}}(A,\gamma))_{/\leftrightarrows_{\mathrm{rb}}} \models \mathrm{AIP}^-$.

Recall that AIP^- requires that at least one of the two considered terms is guarded. In the course of establishing the validity of AIP^-, the following theorem is needed. It characterizes a solution of a guarded recursive specification, or, in other words, a definable process (see Definition 5.7.5). As such, it replaces Theorem 5.7.7 (Processes definable over $\mathrm{BSP}(A)$) in theories with silent steps, and it provides the properties of the process specified by the guarded term in the AIP^- formulation. A transition system is called *divergence free* if and only if from no state of the system, an infinite series of consecutive τ-steps can be taken. An equivalence class of transition systems under rooted branching bisimilarity, i.e., a process in the current context, is divergence free if and only if at least one element is.

Theorem 8.8.9 (Processes definable over $\mathrm{TCP}_\tau(A,\gamma)$**)** A process in the algebra of processes $\mathbb{P}(\mathrm{TCP}_{\tau,\mathrm{rec}}(A,\gamma))_{/\leftrightarrows_{\mathrm{rb}}}$ is definable over $\mathrm{TCP}_\tau(A,\gamma)$ if and only if it contains a transition system as an element that is both finitely branching and divergence free.

The proof of this result can be found in (Baeten *et al.*, 1987b). It should be stressed that this theorem states that a transition system is needed that has both properties at the same time. It is not so that a process is definable if and only if it is finitely branching and divergence free. The latter would allow that some transition systems in the equivalence class are finitely branching and some divergence free but that no single transition system is both finitely branching and divergence free. Consider for example the unguarded recursive specification, using the k-fold action prefix of Notation 4.6.6, that has equations $X_n = \tau.X_{n+1} + \sum_{0\le i\le n} a^i 1$ for each natural number n. The transition system for X_0 obtained by the operational rules is finitely branching but not divergence free, while after removing all τ-steps after the initial one a rooted branching bisimilar transition system is obtained that is divergence free but not finitely branching. The process containing among others these two transition systems is therefore finitely branching and divergence free but it is *not* definable over $\mathrm{TCP}_\tau(A,\gamma)$.

https://doi.org/10.1017/CBO9781139195003.012 Published online by Cambridge University Press

Let us take a closer look at transition systems with divergence that cannot be removed under (rooted) branching bisimilarity. As an example, consider the *unguarded* recursive specification $E = \{X_n = \tau.X_{n+1} + a^n 1 \mid n \in \mathbf{N}\}$. The transition system corresponding to recursion variable X_0 has divergence, and none of the τ-steps can be removed under branching bisimilarity, all states being different. Nevertheless, the process containing the transition system of X_0 is expressible in $\mathrm{BSP}_{\tau,\mathrm{rec}}(A)$ (and hence in $\mathrm{TCP}_{\tau,\mathrm{rec}}(A, \gamma)$), as defined in Definition 5.7.2 (Expressiveness). This is demonstrated by recursive specification E. However, the process is not *definable* over $\mathrm{BSP}_\tau(A)$ or $\mathrm{TCP}_\tau(A, \gamma)$, as this would contradict Theorem 8.8.9 above. This means that it is not the solution of a guarded recursive specification over the corresponding signature (see again (Baeten *et al.*, 1987b)). What can be done however is the following. Consider the guarded recursive specification $F = \{Y_n = i.X_{n+1} + a^n 1 \mid n \in \mathbf{N}\}$, where $i \in A$ is some atomic action. This recursive specification is guarded, so variable Y_0 has a unique solution, in the form of a definable process. Now process $\mu X_0.E$ can be obtained as $\tau_{\{i\}}(\mu Y_0.F)$, i.e., the process specified by unguarded specification E can be obtained as an abstraction applied to a definable process. It turns out that abstraction applied to definable processes is a powerful concept. Abstraction applied to *finitely* definable processes already allows to define *all countable computable* processes. Note that, when assuming that τ is an allowed transition label, the definition of a (countable) computable process as defined in Definition 5.7.1 carries over to the current setting. A process in the current context is an equivalence class of transition systems under rooted branching bisimilarity, but this results in the same set of (countable) computable processes. Based on Theorem 5.7.3 (Countable computable transition systems), this implies that all countable computable processes can be specified by means of $\mathrm{BSP}_{\tau,\mathrm{rec}}(A)$-terms. The following theorem states the universality of theory $\mathrm{TCP}_\tau(A, \gamma)$, meaning that every countable computable process can be defined via a finite expression over the signature of that theory with finite guarded recursion. A proof of the theorem is omitted. The interested reader is referred to (Baeten *et al.*, 1987b).

Theorem 8.8.10 (Universality of $\mathrm{TCP}_\tau(A, \gamma)$**)** Let p be a countable computable process, and $\mathrm{BSP}_{\tau,\mathrm{rec}}(A)$-term t the term specifying p. Then, there is a (finite) set of actions $I \subseteq A$ and a finite guarded recursive specification E over $\mathrm{TCP}_\tau(A, \gamma)$ with recursion variable X, such that $(\mathrm{TCP}_{\tau,\mathrm{rec}} + \mathrm{RSP})(A, \gamma) \vdash \tau_I(\mu X.E) = t$.

Recall from the previous section that divergence is tightly coupled to the notion of fairness. The Fair-Iteration Axiom given in the previous section

https://doi.org/10.1017/CBO9781139195003.012 Published online by Cambridge University Press

expresses the assumption that τ-prefix iteration behaves fairly and divergence does not occur. It is therefore interesting to consider fairness in the current context with general recursion. Consider the following example.

Example 8.8.11 (Tossing a coin) Suppose a statistician carries out an experiment: he tosses a coin until head comes up. Assuming that the probability of tossing heads is between 0 and 1 (so, not equal to 0 or to 1), this process can be described by the following recursive equation.

$$S = \mathit{toss}.(\tau.\mathit{tail}.S + \tau.\mathit{head}.1).$$

In this specification, a non-deterministic choice is used to indicate that the choice cannot be influenced by another process (for instance, by blocking one alternative, or only offering a synchronization with one alternative). It is unclear at what point the outcome is determined: presumably after the moment the coin is thrown into the air, but before the moment the outcome is observed. The moment of choice itself cannot be observed, and this is why silent steps are used. In Section 11.2, a theory with probabilities is discussed that allows for a better description of examples in which probabilities play a role.

Assume now that the experiment is carried out in a closed room. Standing outside, an observer cannot observe anything occurring. However, if head comes up, the statistician yells: 'Head!', which means that the outside observer can 'observe' the *head* action, whereas actions from $I = \{\mathit{toss}, \mathit{tail}\}$ are still hidden from the observer. Thus, the observer observes the process $\tau_I(S)$. Because the coin is *fair*, after a number of tails (zero or more), head will come up. So, following this intuition, one would expect the following equality:

$$\tau_I(S) = \tau.\mathit{head}.1,$$

saying that after some internal activity *head* is observed. Figure 8.11 shows that the transition system associated with S where all actions from I are replaced by the silent step and the transition system associated with $\tau.\mathit{head}.1$ are indeed rooted branching bisimilar.

The question now arises whether the expected equality can also be derived from the axioms of the equational theories considered so far. The Fair-Iteration Axiom of the previous section only considers a τ-loop consisting of a single τ action, and not a cycle consisting of more than one τ-step. This axiom cannot be applied to prove the above equality. In fact, none of the axioms considered so far allows the derivation of the equality. Still, it is preferable to be able to derive this identity equationally.

The Fair-Iteration Axiom can be rephrased in a setting with recursion in the form of the following conditional axiom (see also Section 5.3). The axiom is

https://doi.org/10.1017/CBO9781139195003.012 Published online by Cambridge University Press

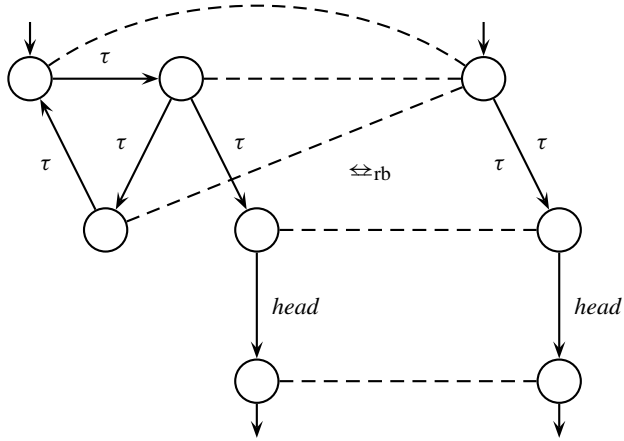


Fig. 8.11. Statistician.

given as a deduction rule, with x and y process terms, $i \in A$ an action, and $I \subseteq A$ some set of actions.

$$\frac{x = i.x + y,\ i \in I}{\tau.\tau_I(x) = \tau.\tau_I(y)} \qquad (\mathrm{KFAR}_1^{\mathrm{b}}).$$

Notice that in the equation $x = i.x + y$, the occurrence of recursion variable x is guarded, so the process x is uniquely defined and it makes sense to write $\tau_I(x)$.

In general, when there is a cycle of internal steps i_k of length n, conditional axioms, one for each n, can be formulated as follows.

$$\frac{\begin{array}{l} x_1 = i_1.x_2 + y_1 \\ x_2 = i_2.x_3 + y_2 \\ \vdots \\ x_{n-1} = i_{n-1}.x_n + y_{n-1} \\ x_n = i_n.x_1 + y_n \\ i_1, i_2, \cdots, i_n \in I \cup \{\tau\}, \text{ with at least one } i_k \not\equiv \tau \end{array}}{\tau.\tau_I(x_1) = \tau.(\tau_I(y_1) + \tau_I(y_2) + \cdots + \tau_I(y_n))} \qquad (\mathrm{KFAR}_n^{\mathrm{b}}).$$

The i_k actions in the equations for x_k form a cycle; the y_k specify the behaviors that leave the cycle. Figure 8.12 illustrates $\mathrm{KFAR}_4^{\mathrm{b}}$. Note that some of the i_k may be equal to τ, but not all of them, since that would make the specification of the x_k unguarded.

The given deduction rules are collectively called *Koomen's Fair Abstraction Rules*, abbreviated as $\mathrm{KFAR}^{\mathrm{b}}$. The 'b' in this name indicates that this

https://doi.org/10.1017/CBO9781139195003.012 Published online by Cambridge University Press

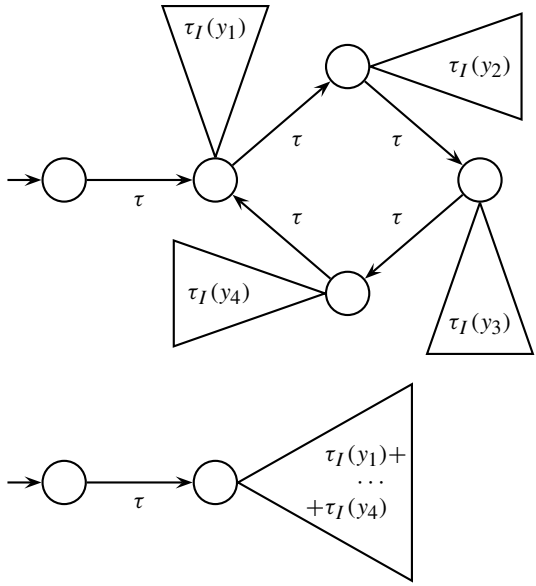


Fig. 8.12. An illustration of $\mathrm{KFAR}^{\mathrm{b}}$.

formulation holds in the context of (rooted) branching bisimilarity. The original formulation of Koomen's Fair Abstraction Rules only works in the context of another semantic equivalence called weak bisimulation (see (Baeten *et al.*, 1987b)). $\mathrm{KFAR}^{\mathrm{b}}$ says that, in abstracting from a set of internal actions, eventually (i.e., after performing a number of internal steps) an external step will be chosen (unless no such steps exist).

Theorem 8.8.12 (Validity of $\mathrm{KFAR}^{\mathrm{b}}$**)** Principle $\mathrm{KFAR}^{\mathrm{b}}$ is valid in both the term models with and without recursion constants: $\mathbb{P}(\mathrm{TCP}_{\tau}(A,\gamma))_{/\leftrightarrows_{\mathrm{rb}}} \models \mathrm{KFAR}^{\mathrm{b}}$ and $\mathbb{P}(\mathrm{TCP}_{\tau,\mathrm{rec}}(A,\gamma))_{/\leftrightarrows_{\mathrm{rb}}} \models \mathrm{KFAR}^{\mathrm{b}}$.

The proof of this result is straightforward and left to the reader. The theorem states that the collective set of $\mathrm{KFAR}^{\mathrm{b}}$ deduction rules is valid in the standard term models. However, it is interesting to observe that the individual rules are independent. For any n, there is a model for $\mathrm{TCP}_{\tau,\mathrm{rec}}(A,\gamma)$ that validates $\mathrm{KFAR}^{\mathrm{b}}_{n+1}$ but not $\mathrm{KFAR}^{\mathrm{b}}_{n}$. A proof of this fact is outside the scope of this text, but can be found in (Baeten & Weijland, 1990).

Example 8.8.13 (Tossing a coin) Consider the following description of the statistician tossing a coin until head comes up, that is equivalent to the one used in Example 8.8.11:

https://doi.org/10.1017/CBO9781139195003.012 Published online by Cambridge University Press

$$\begin{aligned} S &= toss.S' + 0, \\ S' &= \tau.S'' + \tau.head.1, \\ S'' &= tail.S + 0. \end{aligned}$$

Recall that $I = \{toss, tail\}$. Then, using $\mathrm{KFAR}^{\mathrm{b}}$, the following derivation can be obtained. Note that communication does not play a role, so γ can be chosen arbitrarily.

$$\begin{aligned} &(\mathrm{TCP}_{\tau,\mathrm{rec}} + \mathrm{KFAR}^{\mathrm{b}})(A, \gamma) \vdash \\ &\quad \tau.\tau_I(S') = \tau.(\tau_I(\tau.head.1) + \tau_I(0) + \tau_I(0)) \\ &\qquad\qquad = \tau.\tau.head.1 \\ &\qquad\qquad = \tau.head.1. \end{aligned}$$

It then follows easily that

$$\begin{aligned} &(\mathrm{TCP}_{\tau,\mathrm{rec}} + \mathrm{KFAR}^{\mathrm{b}})(A, \gamma) \vdash \\ &\quad \tau_I(S) = \tau_I(toss.S') = \tau.\tau_I(S') = \tau.\tau.head.1 = \tau.head.1, \end{aligned}$$

which proves the desired result.

Example 8.8.14 (Throwing a die) The statistician of Example 8.8.11 now decides to throw a die until six comes up. This is described by

$$\begin{aligned} S_2 = throw.(&\tau.one.S_2 + \tau.two.S_2 + \tau.three.S_2 \\ &+ \tau.four.S_2 + \tau.five.S_2 + \tau.six.1). \end{aligned}$$

Again, the experiment is carried out in a room, and the observer outside can only hear the yell 'Six!'. Abstracting from actions $I = \{throw, one, two, three, four, five\}$, one would expect that the identity $\tau_I(S_2) = \tau.six.1$ is derivable from the equational theory. However, the proof principle $\mathrm{KFAR}^{\mathrm{b}}$ cannot be used directly to prove this expected result. (Try it.)

The reason that $\mathrm{KFAR}^{\mathrm{b}}$ cannot be applied in the above example is that the structure of the recursive equation does not lead to a simple cycle, as illustrated in Figure 8.13. Using a trick, based on renaming of actions (see Section 6.7), it is possible to apply $\mathrm{KFAR}^{\mathrm{b}}$ nonetheless, see Exercise 8.8.5. This trick can be used in some cases to extend the applicability of the proof principle, but also fails for more complicated structures of transition systems (although even then, with a more general theory of renaming, $\mathrm{KFAR}^{\mathrm{b}}$ is sufficient (Baeten & Weijland, 1990; Vaandrager, 1986)). Therefore, it is useful to have a version of $\mathrm{KFAR}^{\mathrm{b}}$ that is applicable to arbitrary so-called *clusters* of internal steps: the *Cluster Fair Abstraction Rule* $\mathrm{CFAR}^{\mathrm{b}}$.

Definition 8.8.15 (Cluster) Let E be a recursive specification, and let $I \subseteq A$. A subset C of variables from E is called a *cluster of I in $\mu X.E$* if variable X of E is in C and if the following condition holds: for all $Z \in C$, there

https://doi.org/10.1017/CBO9781139195003.012 Published online by Cambridge University Press

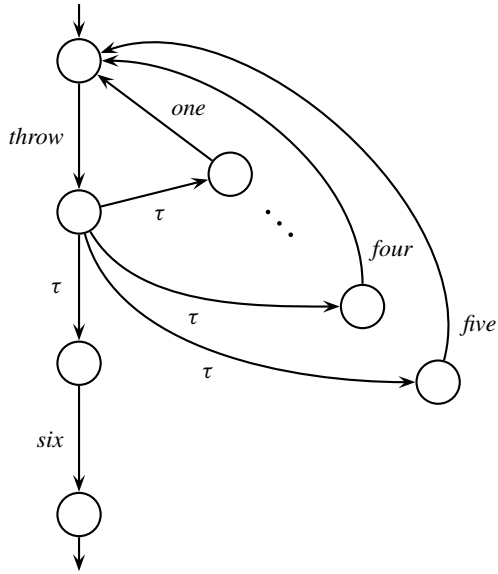


Fig. 8.13. The statistician throwing dice.

exist actions $i_1, \cdots, i_m \in I \cup \{\tau\}$, recursion variables $X_1, \cdots, X_m \in C$, and recursion variables $Y_1, \cdots, Y_n \notin C$ for some $m \geq 1$ and $n \geq 0$ such that the equation for Z in E is of the form

$$Z = \sum_{k=1}^{m} i_k.X_k + \sum_{j=1}^{n} Y_j.$$

The variables Y_j in this equation are called the *exits* of Z and denoted $U(Z)$. Furthermore, the exit set of the cluster C is $U(C) = \bigcup_{Z \in C} U(Z)$. A cluster is called *conservative* if every exit $Y \in U(C)$ is accessible from every variable in the cluster by doing a number of steps from $I \cup \{\tau\}$ to a cluster-variable which has exit Y.

Example 8.8.16 (Cluster) Reconsider process S_2 of Example 8.8.14. This process can be defined by the following guarded recursive specification E:

$$\begin{aligned}
S_2 &= \mathit{throw}.X_0 + 0\\
X_0 &= \tau.X_1 + \tau.X_2 + \tau.X_3 + \tau.X_4 + \tau.X_5 + X_6\\
X_1 &= \mathit{one}.S_2 + 0\\
X_2 &= \mathit{two}.S_2 + 0\\
X_3 &= \mathit{three}.S_2 + 0\\
X_4 &= \mathit{four}.S_2 + 0
\end{aligned}$$

https://doi.org/10.1017/CBO9781139195003.012 Published online by Cambridge University Press

$$X_5 = \mathit{five}.S_2 + 0$$
$$X_6 = \tau.\mathit{six}.1.$$

Then, $C = \{S_2, X_0, X_1, X_2, X_3, X_4, X_5\}$ is a conservative cluster of $I = \{\mathit{throw}, \mathit{one}, \mathit{two}, \mathit{three}, \mathit{four}, \mathit{five}\}$ with exit X_6 in any of the processes $\mu X.E$ with X in C. Note that the 0 summands in the equations correspond to empty summations. They are included to illustrate how the equations fit Definition 8.8.15.

Definition 8.8.17 (CFAR$^{\mathrm{b}}$) Let E be a guarded recursive specification in which the recursion variable X occurs and let $I \subseteq A$. Let C be a finite conservative cluster of I in $\mu X.E$ and let U be the set of exits from the cluster C. Then, the following identity is derivable.

$$\tau.\tau_I(X) = \tau.\left(\sum_{Y \in U} \tau_I(Y)\right) \qquad (\mathrm{CFAR}^{\mathrm{b}}).$$

Note that CFAR$^{\mathrm{b}}$ is, like KFAR$^{\mathrm{b}}$, a conditional axiom. It is valid for the standard term model of theory $\mathrm{TCP}_{\tau,\mathrm{rec}}(A, \gamma)$ (Baeten & Weijland, 1990; Vaandrager, 1986).

Example 8.8.18 (Application of CFAR$^{\mathrm{b}}$**)** Return to Example 8.8.16. The given cluster C is finite. Consider the process specified by X_0. By CFAR$^{\mathrm{b}}$, it follows that

$$(\mathrm{TCP}_{\tau,\mathrm{rec}} + \mathrm{CFAR}^{\mathrm{b}})(A, \gamma) \vdash \tau.\tau_I(X_0) = \tau.\tau_I(X_6) = \tau.\mathit{six}.1.$$

This result easily leads to the result that

$$(\mathrm{TCP}_{\tau,\mathrm{rec}} + \mathrm{CFAR}^{\mathrm{b}})(A, \gamma) \vdash \tau_I(S_2) = \tau.\mathit{six}.1,$$

which is the desired result, as explained in Example 8.8.14.

Exercises

8.8.1 Recall the specification and calculations for a two-place buffer from Section 7.6. A two-place buffer with input channel i and output channel o is given by the following equations:

$$\mathit{Buf}2 = 1 + \sum_{d \in D} i?d.B_d \quad \text{and, for all } d \in D,$$
$$B_d = o!d.\mathit{Buf}2 + \sum_{e \in D} i?e.o!d.B_e.$$

Assuming that $\mathit{Buf}1_{pq}$ is a one-place buffer with input port p and output port q, i.e.,

$$\mathit{Buf}1_{pq} = 1 + \sum_{d \in D} p?d.q!d.\mathit{Buf}1_{pq},$$

https://doi.org/10.1017/CBO9781139195003.012 Published online by Cambridge University Press

and assuming the standard communication function γ_S (see Definition 7.3.1), prove that

$$(\mathrm{TCP}_{\tau,\mathrm{rec}} + \mathrm{RSP})(A, \gamma_S) \vdash \\ Buf2 = \tau_I(\partial_H(Buf1_{il} \parallel Buf1_{lo})),$$

where $H = \{l!d, l?d \mid d \in D\}$ and $I = \{l\mathrm{c}d \mid d \in D\}$.

8.8.2 Prove that the recursive specification $\{X = i.\tau_{\{j\}}(Y),\ Y = j.\tau_{\{i\}}(X)\}$ has more than one solution in the standard term model of theory $\mathrm{TCP}_{\tau,\mathrm{rec}}(A, \gamma)$ based on rooted branching bisimilarity.

8.8.3 Try to define the notion of guardedness for recursive specifications that, in contrast to what is allowed by Definition 8.8.3 (Guardedness in $\mathrm{TCP}_{\tau,\mathrm{rec}}(A, \gamma)$), may contain abstraction operators. As usual, the definition should guarantee that every guarded recursive specification has a unique solution in the standard term model, based on rooted branching bisimilarity in the current context. Consider only finite specifications.

8.8.4 Show that the conclusion of the $\mathrm{KFAR}^{\mathrm{b}}_n$ deduction rule can equivalently be formulated as follows:

$$\tau_I(x_1) = \tau_I(y_1) + \tau.(\tau_I(y_1) + \cdots + \tau_I(y_n)).$$

8.8.5 Let $I \subseteq A$ be some set of actions, and let $t \in A$ be some action. Define a renaming function (see Section 6.7) f by $f(a) = t$, if $a \in I$, and $f(a) = a$, otherwise. The renaming operator ρ_f is denoted t_I and called the *pre-abstraction* operator. Prove for all closed $\mathrm{TCP}_\tau(A, \gamma)$-terms p that

$$\mathrm{TCP}_\tau(A, \gamma) \vdash \tau_I(p) = \tau_{\{t\}}(t_I(p)),$$

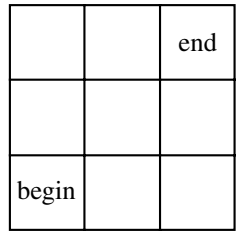
whenever t does not occur in p.

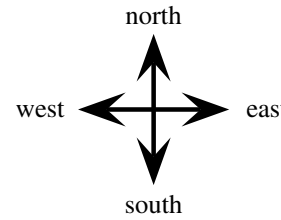
Assume this equation also holds for recursively defined processes and reconsider Example 8.8.14 (Throwing a die). Use this equation to derive $\tau_I(S_2) = \tau.six.1$, using only the axioms of $\mathrm{TCP}_{\tau,\mathrm{rec}}(A, \gamma)$ and $\mathrm{KFAR}^{\mathrm{b}}$.

8.8.6 Show that $\mathrm{KFAR}^{\mathrm{b}}$ can be derived from $\mathrm{CFAR}^{\mathrm{b}}$.

8.8.7 Suppose someone starts a random walk in the lower left-hand corner of the 3×3 grid shown below. When arriving in the upper right-hand corner, the person stops walking. The process P describing this random walk has actions *begin*, *north*, *west*, *south*, *east*, *end*. Give a recursive specification for P. Let $I = \{north, west, south, east\}$ and calculate $\tau_I(P)$ using $\mathrm{CFAR}^{\mathrm{b}}$. Does this result conform to your intuition?

https://doi.org/10.1017/CBO9781139195003.012 Published online by Cambridge University Press





8.8.8 Formulate a variant of the $\mathrm{KFAR}^{\mathrm{b}}$ principle that is suitable for the abstraction of unstable divergence, as treated in Exercise 8.7.4. Use the resulting rule to establish the identity desired in Example 8.8.11 (Tossing a coin).

8.9 Verification of the ABP and queues revisited

Consider again the Alternating-Bit Protocol from Section 7.8. In that section, it was already mentioned that the process $\partial_H(S \parallel K \parallel L \parallel R)$ after abstraction of internal actions, satisfies the specification of the one-place buffer $Buf1_{io}$. In this chapter, these claims are backed with a formal derivation of the following theorem.

Theorem 8.9.1 (Alternating-Bit Protocol) The Alternating-Bit Protocol is a *correct* communication protocol, i.e.,

$$((\mathrm{TCP}_\tau + \mathrm{HA})_{\mathrm{rec}} + \mathrm{CFAR}^{\mathrm{b}} + \mathrm{RSP})(A, \gamma_S) \vdash \tau_I(\partial_H(S \parallel K \parallel L \parallel R)) = Buf1_{io},$$

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

Proof In Section 7.8, the following recursive specification has been derived for the process $\partial_H(S \parallel K \parallel L \parallel R)$, using among others the Handshaking Axiom HA. The specification uses the recursion variables X, $X1_d$, $X2_d$, Y, $Y1_d$, and $Y2_d$ (for each $d \in D$), with process X equal to $\partial_H(S \parallel K \parallel L \parallel R)$:

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

https://doi.org/10.1017/CBO9781139195003.012 Published online by Cambridge University Press

As the next step, for the process $X1_d$, an alternative recursive specification is given using additional recursion variables $Z_1, \cdots, Z_6$. The reason for doing so is that the above recursive specification does not fit well with the format required for applying CFAR$^{\mathrm{b}}$. The equivalence of the two recursive specifications can be shown in a straightforward way.

$$\begin{aligned}
X1_d &= sk?d0.Z_1,\\
Z_1 &= t.Z_2 + t.kr?d0.o!d.X2_d,\\
Z_2 &= kr?\bot.Z_3,\\
Z_3 &= rl?1.Z_4,\\
Z_4 &= t.Z_5 + t.Z_6,\\
Z_5 &= ls?1.X1_d,\\
Z_6 &= ls?\bot.X1_d.
\end{aligned}$$

Then, $\{X1_d, Z_1, Z_2, Z_3, Z_4, Z_5, Z_6\}$ is a finite conservative cluster of I, with exit $t.kr?d0.o!d.X2_d$. (Note that, when strictly following Definition 8.8.15 (Cluster), the exit should have been defined using an additional recursion variable.) From CFAR$^{\mathrm{b}}$, it then follows that

$$\begin{aligned}
&(\mathrm{TCP}_{\tau,\mathrm{rec}} + \mathrm{CFAR}^{\mathrm{b}})(A, \gamma_S) \vdash\\
&\quad \tau_I(X1_d) = \tau.\tau_I(t.kr?d0.o!d.X2_d) = \tau.o!d.\tau_I(X2_d).
\end{aligned}$$

So, the first cluster of internal steps, as visualized in Figure 7.9, has disappeared. The same procedure is followed for the cluster around $X2_d$. The alternative recursive specification is the following:

$$\begin{aligned}
X2_d &= rl?0.Z_1,\\
Z_1 &= t.Z_2 + t.ls?0.Y,\\
Z_2 &= ls?\bot.Z_3,\\
Z_3 &= sk?d0.Z_4,\\
Z_4 &= t.Z_5 + t.Z_6,\\
Z_5 &= kr?d0.X2_d,\\
Z_6 &= kr?\bot.X2_d.
\end{aligned}$$

Again, from the fact that $\{X2_d, Z_1, Z_2, Z_3, Z_4, Z_5, Z_6\}$ is a finite conservative cluster of I, it follows that:

$$\begin{aligned}
&(\mathrm{TCP}_{\tau,\mathrm{rec}} + \mathrm{CFAR}^{\mathrm{b}})(A, \gamma_S) \vdash\\
&\quad \tau_I(X2_d) = \tau.\tau_I(t.ls?d0.Y) = \tau.\tau_I(Y).
\end{aligned}$$

Combining these results leads to

https://doi.org/10.1017/CBO9781139195003.012 Published online by Cambridge University Press

$$
\begin{aligned}
&(\mathrm{TCP}_{\tau,\mathrm{rec}} + \mathrm{CFAR}^{\mathrm{b}})(A, \gamma_S) \vdash \\
&\quad \tau_I(X) = 1 + \sum_{d\in D} i?d.\tau_I(X1_d) \\
&\qquad\quad = 1 + \sum_{d\in D} i?d.\tau.o!d.\tau_I(X2_d) \\
&\qquad\quad = 1 + \sum_{d\in D} i?d.o!d.\tau.\tau_I(Y) \\
&\qquad\quad = 1 + \sum_{d\in D} i?d.o!d.\tau_I(Y).
\end{aligned}
$$

In the same way, one can derive

$$
\begin{aligned}
&(\mathrm{TCP}_{\tau,\mathrm{rec}} + \mathrm{CFAR}^{\mathrm{b}})(A, \gamma_S) \vdash \\
&\quad \tau_I(Y) = 1 + \sum_{d\in D} i?d.o!d.\tau_I(X).
\end{aligned}
$$

Now, consider the guarded recursive specification:

$$
\begin{aligned}
W_1 &= 1 + \sum_{d\in D} i?d.o!d.W_2, \\
W_2 &= 1 + \sum_{d\in D} i?d.o!d.W_1.
\end{aligned}
$$

It is straightforward that the equations for W_1 and W_2 can be derived from $(\mathrm{TCP}_{\tau,\mathrm{rec}} + \mathrm{CFAR}^{\mathrm{b}})(A, \gamma_S)$ for both the following substitutions for W_1 and W_2:

$$
\begin{aligned}
W_1 &\mapsto \tau_I(X), \\
W_2 &\mapsto \tau_I(Y),
\end{aligned}
\qquad \text{and} \qquad
\begin{aligned}
W_1 &\mapsto \mathit{Buf}1_{io}, \\
W_2 &\mapsto \mathit{Buf}1_{io}.
\end{aligned}
$$

Using RSP, it follows that

$$
(\mathrm{TCP}_{\tau,\mathrm{rec}} + \mathrm{CFAR}^{\mathrm{b}} + \mathrm{RSP})(A, \gamma_S) \vdash \tau_I(X) = \mathit{Buf}1_{io}.
$$

□

Observe that the use of $\mathrm{CFAR}^{\mathrm{b}}$ in the verification of the Alternating-Bit Protocol means in fact that the choice made by the channels is fair. Doing this, the possibility that any of the channels is completely defective is excluded.

To finish this section, consider again the two specifications of the process queue proven equal in Proposition 7.7.4 (Queues). Here, another specification of the unbounded FIFO queue is given, that uses abstraction in an essential way. The idea behind the following specification is that two queues chained together in sequence behave exactly like one single queue, as long as the internal communications are hidden. The idea is illustrated in Figure 8.14. Here, Q^{pq} stands for the queue with input port p and output port q.

Define $H_p = \{p?d, p!d \mid d \in D\}$ and $I_p = \{p\mathfrak{?}d \mid d \in D\}$ for $p = i, l, o$. Consider the following recursive specification of Q^{io}. This specification has six variables Q^{pq}, for each pair of distinct values from $\{i, l, o\}$, and Q^{pq} uses auxiliary port r, such that $\{p, q, r\} = \{i, l, o\}$.

https://doi.org/10.1017/CBO9781139195003.012 Published online by Cambridge University Press

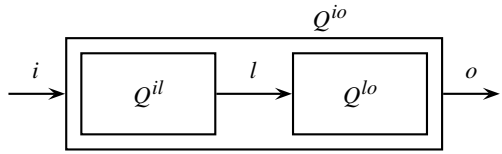


Fig. 8.14. Queue with abstraction.

$$
\begin{aligned}
Q^{io} &= 1 + \sum_{d\in D} i?d.\tau_{I_l}(\partial_{H_l}(Q^{il} \parallel l!d.Q^{lo})),\\
Q^{il} &= 1 + \sum_{d\in D} i?d.\tau_{I_o}(\partial_{H_o}(Q^{io} \parallel l!d.Q^{ol})),\\
Q^{lo} &= 1 + \sum_{d\in D} l?d.\tau_{I_i}(\partial_{H_i}(Q^{li} \parallel o!d.Q^{io})),\\
Q^{ol} &= 1 + \sum_{d\in D} o?d.\tau_{I_i}(\partial_{H_i}(Q^{oi} \parallel l!d.Q^{il})),\\
Q^{li} &= 1 + \sum_{d\in D} l?d.\tau_{I_o}(\partial_{H_o}(Q^{lo} \parallel i!d.Q^{oi})),\\
Q^{oi} &= 1 + \sum_{d\in D} o?d.\tau_{I_l}(\partial_{H_l}(Q^{ol} \parallel i!d.Q^{li})).
\end{aligned}
$$

Note that this is not a guarded recursive specification. Nevertheless, it has a unique solution in the term model that coincides with the solution of the specification $Queue1$ from Proposition 7.7.4 (Queues). This result is not proven here. The reader interested in a proof is referred to (Van Glabbeek & Vaandrager, 1993). The proof in (Van Glabbeek & Vaandrager, 1993) uses auxiliary operators. It is unknown whether a direct proof using only the framework developed in this chapter exists.

Exercises

8.9.1 Let the processes S, K, and R, and the set of blocked actions H be defined as in Exercise 7.8.1. Derive a recursive equation for the process $\tau_I(\partial_H(S \parallel K \parallel R))$, where $I = \{s?x, r?x \mid x \in D \cup \{ack\}\} \cup \{t\}$. Is this communication protocol correct?

8.10 Bibliographical remarks

The silent step τ was introduced in CCS, see (Milner, 1980), to abstract from unobservable behavior. Notation τ is used in that work to stand for the residual trace of a communication. As a semantic equivalence, originally, the notion of (rooted) weak bisimilarity was used both in CCS-style (Milner, 1989) and in ACP-style process algebras (Bergstra & Klop, 1985).

The present treatment is based on the semantic equivalence of (rooted)

https://doi.org/10.1017/CBO9781139195003.012 Published online by Cambridge University Press

branching bisimilarity from (Van Glabbeek & Weijland, 1996; Van Glabbeek & Weijland, 1989). Note that the definition presented here differs from the original definition. The current definition, leading to the same equivalence, was called *semi*-branching bisimilarity earlier, and makes the proof of Theorem 8.2.5 (Equivalence) easier (Basten, 1996). A comparison between branching and weak bisimilarity can be found in (Van Glabbeek, 1994). Branching bisimilarity was also used in (Baeten & Weijland, 1990).

The abstraction operator τ_I was introduced in (Bergstra & Klop, 1985). The axiomatization of CSP's external choice is new here. Earlier axiomatizations can be found in (Brookes, 1983; D'Argenio, 1995; Van Glabbeek, 1997). For a definition in the absence of the empty process, see (Baeten & Bravetti, 2006).

The material on divergence is based on (Bergstra *et al.*, 1987), see also (Baeten & Weijland, 1990). Other references covering divergence are (Aceto & Hennessy, 1992; Walker, 1990). A reference to fair iteration is (Bergstra *et al.*, 2001). For bisimilarity with explicit divergence, see (Van Glabbeek, 1993), and further work in (Van Glabbeek *et al.*, 2008). The notion of catastrophic divergence is due to (Brookes *et al.*, 1984).

The material on recursion is based on (Baeten *et al.*, 1987b). Koomen's Fair Abstraction Rule was first applied by C.J. Koomen in a formula manipulation package based on CCS (see (Koomen, 1985)) and first formulated as a conditional equation in (Bergstra & Klop, 1986a). The present formulation of $\mathrm{KFAR}^{\mathrm{b}}$ and $\mathrm{CFAR}^{\mathrm{b}}$ for branching bisimilarity is from (Baeten & Weijland, 1990). The generalization of KFAR to CFAR is from (Vaandrager, 1986). Other references concerning fairness are (Francez, 1986; Parrow, 1985).

The verification of the Alternating-Bit Protocol is from (Bergstra & Klop, 1986c). For further information on algebraic verification, see (Groote & Reniers, 2001). The specification of the queue is from (Bergstra & Klop, 1986b), see also (Van Glabbeek & Vaandrager, 1993).

9
Timing

9.1 Introduction

The process theories introduced so far describe the main features of imperative concurrent programming without the explicit mention of time. Implicitly, time is present in the interpretation of many of the operators introduced before. In the process $a.x$, the action a must be executed *before* the execution of process x. The process theories introduced so far allow for the description of the ordering of actions relative to each other. This way of describing the execution of actions through time is called *qualitative time*. Many systems though rely on time in a more quantitative way.

Consider for example the following caller process. A caller takes a phone off the hook. If she hears a certain tone, she dials some number. It does not matter which one. If she does not hear the tone, she puts the phone back on the hook. After dialing the number, the caller waits some time for the other side to pick up the phone. After some conversation, the caller puts the phone back on the hook. In case the call is not answered within some given time, the caller gives up and also puts the phone back on the hook.

To be able to describe such systems in process theory in the same framework as untimed systems, many process theories have been extended with a quantitative notion of timing. In extending the untimed process theories with timing a number of fundamental choices have to be made with respect to the nature of the time domain, the way time appears syntactically in the equational theory, and the way time is incorporated semantically.

Linear time versus branching time Different time domains that are used in modeling real-life applications are the natural numbers and the non-negative real numbers. These time domains have in common that any two moments in time can be compared using some total ordering $\leq$ on that time

https://doi.org/10.1017/CBO9781139195003.013 Published online by Cambridge University Press

domain. Time domains for which such an ordering is given (and used in comparing moments in time) are called *linear* time domains. In principle, one could also consider a time domain where only some of the moments in time can be compared, e.g., to express the lack of a globally synchronized clock, or to model relativistic space/time. These are the *branching* time domains. The time domains that appear in the process algebra literature, but also in almost any case study, are of the first type. In this chapter, therefore, a *linear*-time process theory is developed.

Discrete versus dense time A difference between the naturals and the non-negative reals is that in the latter between any two reals another one can be found, whereas for the naturals this is not possible. The naturals form an instance of a so-called *discrete* time domain; the non-negative reals are called a *dense* time domain. In this chapter, a *discrete*-time process theory is developed. A reason for selecting this type of time domain is that the relation with the untimed theory is easier to establish. In a discrete time domain, time is divided in so-called *time slices*: actions take place within a certain time slice, and within a certain time slice, ordering of actions is only qualitative. In moving to the next time slice, a special 'tick' event takes place, resembling the notion of a clock tick.

As a remark aside, note that the use of an uncountable, dense time domain such as the non-negative reals would provide means to specify uncountable processes, which is not possible with any of the techniques worked out in detail in this book; for more details, see the discussion and results in Section 5.7.

Absolute versus relative time Depending on the type of applications one wishes to describe using the timed process theory, either the passage of time is described relatively to a global clock, or relative to the previous action. The first way of describing time is called *absolute* time, the latter *relative* time. Process theories where both paradigms are combined also exist in the literature; these are called *parametric*-time process theories. In this chapter, a relative-time process theory is developed.

Timed action execution In the untimed process theory, it has been assumed that actions occur instantaneously. In developing a timed process theory, one has to decide whether actions have a duration or are still considered to occur instantaneously. In this chapter, the execution of an action in the timed process theory is assumed to be instantaneous, or maybe it is better to say that the *observation* of action execution is instantaneous. Thus, the interleaving approach to parallel composition can still be followed, and simultaneous

occurrence of actions is reserved for the description of communication. In a discrete-time theory, action execution within a certain time slice can be dealt with as in untimed process algebra, but in a dense-time theory, a closer look has to be given to simultaneous execution of actions, as actions in unrelated parts of a system might occur at the same moment of time. Then, the usual choice in process algebra is to allow actions to execute consecutively at the same moment of time, rather than the alternative of using so-called multi-actions for such an occurrence.

Time-stamped versus two-phase description Capturing the timing aspects of a process can essentially be done in two ways. One way is to attach to each action a moment in time, or a time slice. This way of describing time is called *time-stamping*. The other way is to denote the passage of time itself in between actions explicitly. This way of describing time is paraphrased as the *two-phase* description of time. To stay as close as possible to the untimed transition systems and the syntax of the process theories in the previous chapters, in this chapter, a two-phase approach is used at the syntactical level of description. On the other hand, semantically, a time-stamped treatment is followed. The latter allows a better treatment of time determinism (to be treated next), and allows easier extensions.

Time determinism versus time non-determinism In the previous chapters, the execution of an action resulted in resolving a possible choice between alternatives. Thus, in a term $a.1 + b.1$, it is possible to execute a and thereby not do b, or to execute b and thereby not do a. As long as neither action has occurred, the choice is not determined. Now it can be the case that a and b are constrained in time; for example, it might happen that a can only occur in the following time slice (after one 'tick' event has occurred), and that b can only occur in the time slice thereafter (after two ticks). Then, when the first tick occurs, no choice is made, as both actions are still possible. This is called *time determinism*: it is not possible that time evolves in different ways. Thus, the tick event is different from an action in this respect. Most timed process algebras adopt the principle of time determinism, but there are exceptions.

Continuing the example, after the first tick, the action a can be executed, thereby disabling b. In this book, it is also allowed that a second tick event occurs, thereby moving to the following time slice, in which the choice for b can be effectuated. The occurrence of the second tick disables the occurrence of a, as the time slice in which it should occur has passed. Thus, the passage of time can disable a choice. This is called *weak time determinism*. When the principle of *strong time determinism* is adopted, on the other hand, then the second tick

cannot be executed, and in this example, necessarily a must occur, and action execution has priority over passage of time. Adopting weak time determinism as is done here adheres to the intuition of alternative composition. Moreover, it makes it easier to describe timeouts.

Interpretation of untimed processes A crucial point in the development of the timed process theory in this chapter is the relation between the untimed process theories of the previous chapters and the timed process theory of this chapter. The view adopted in this book is that the timed process theory should be an equationally conservative ground-extension of the untimed process theory. As a consequence, every untimed process term should be interpreted in the timed setting in a consistent and meaningful way, especially, taking care to respect the identities between untimed processes. Basically, there are two ways of interpreting an untimed process. The first one is to assume all action execution to take place in one and the same time slice. The second one is to assume all action execution to take place arbitrarily dispersed over time though still respecting the ordering of actions as described in the untimed process term. This last interpretation fits better with an engineering discipline where first an untimed model is given, and after verification of the functional behavior, timing information is added to perform some timeliness verification. Hence, in this chapter, a timed process theory is developed that is a conservative ground-extension of the untimed theory under the interpretation that untimed actions can take place at an arbitrary moment in time.

9.2 Timed transition systems

Definition 9.2.1 (Timed transition-system space) In order to extend a transition-system space to take timing into account, both action execution and termination are extended with an extra parameter, a natural number that indicates after how many ticks the action or termination takes place. Moreover, there is an extra predicate $\rightsquigarrow$ that indicates how many ticks are possible from a given state.

A *timed transition-system space* over a set of labels L is a set S of states, equipped with one quaternary relation $\rightarrow$ and two binary relations $\downarrow$ and $\rightsquigarrow$:

(i) $\rightarrow \subseteq S \times \mathbf{N} \times L \times S$ is the set of *transitions*;
(ii) $\downarrow \subseteq S \times \mathbf{N}$ is the set of *terminating* states;
(iii) $\rightsquigarrow \subseteq S \times \mathbf{N}$ is the set of possible *delays*.

The notation $s \ {}_n\!\xrightarrow{a}\ t$ is used for $(s, n, a, t) \in \ \rightarrow$, and means intuitively that from state s, after n ticks, action a can be executed resulting in state t;

https://doi.org/10.1017/CBO9781139195003.013 Published online by Cambridge University Press

next, $s\downarrow_n$ is used for $(s, n) \in \downarrow$, and means that from state s, after n ticks, termination is possible; finally, $s \rightsquigarrow_n$, notation for $(s, n) \in \rightsquigarrow$, means that n ticks are possible from state s. The following implication is assumed to hold: whenever $s \xrightarrow[n]{a} t$ or $s\downarrow_n$, then $s \rightsquigarrow_n$. Note that a state may satisfy other $\rightsquigarrow_n$ predicates in addition to those that are implied by the transitions and termination options. These can be used to specify timing behavior independent of actions or termination.

In the rest of this chapter, assume that $(S, L, \rightarrow, \downarrow, \rightsquigarrow)$ is a timed transition-system space. Each state $s \in S$ can be identified with a timed transition system that consists of all states and transitions reachable from s. The notion of reachability is defined by generalizing the transition relations.

Definition 9.2.2 (Reachability) The reachability relation $\rightarrow^* \subseteq S \times S$ is defined inductively as follows:

(i) $s \rightarrow^* s$ for each $s \in S$;
(ii) for all $s, t, u \in S$, $n \in \mathbf{N}$, and $a \in L$, if $s \rightarrow^* t$ and $t \xrightarrow[n]{a} u$, then $s \rightarrow^* u$.

A state $t \in S$ is said to be *reachable* from state $s \in S$ if and only if $s \rightarrow^* t$.

Definition 9.2.3 (Timed transition system) For each state $s \in S$, the *timed transition system* associated with s consists of all states reachable from s, and has the transitions and terminating states induced by the timed transition-system space. State s is called the *initial state* or *root* of the timed transition-system associated with s. Usually, 'timed transition system s' is used to refer to the timed transition system *associated with* s.

The graphical representation of timed transition systems is similar to the graphical representation of transition systems in the previous chapters. The arrows and terminations have a natural number as an additional label. The $\rightsquigarrow$ predicates are omitted if these are implied by other arrows or terminations, and are denoted with dotted arrows annotated with natural numbers otherwise.

Example 9.2.4 (Timed transition system of the caller process) The timed transition system in Figure 9.1 represents the caller process described in the introduction to this chapter. From state 0, there is a transition for every $n \in \mathbf{N}$; state 4 has a transition for every $m \in \mathbf{N}$.

Example 9.2.5 (Timed transition system) Figure 9.2 shows another example of a timed transition system. In the initial state, it has among others an

https://doi.org/10.1017/CBO9781139195003.013 Published online by Cambridge University Press

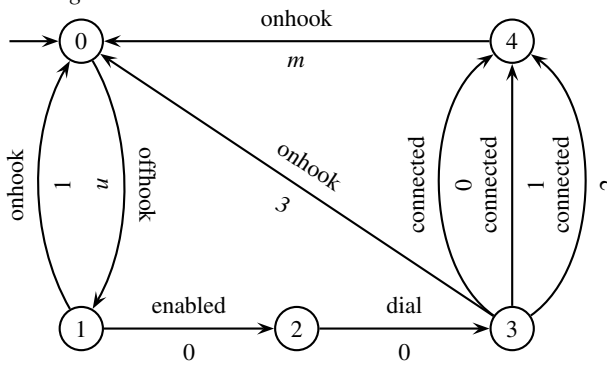


Fig. 9.1. An example of a timed transition system: the caller process.

option to allow the passage of time for two time units, after which time cannot progress any further, the process cannot terminate successfully, and it can no longer continue anymore with any action (i.e., it deadlocks).

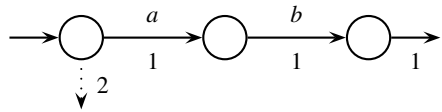


Fig. 9.2. Another example of a timed transition system.

Definition 9.2.6 (Timed bisimilarity) A binary relation R on the set of states S of a timed transition-system space is a *timed bisimulation relation* if and only if the following transfer conditions hold:

(i) for all states $s, t, s' \in S$, whenever $(s, t) \in R$ and $s \xrightarrow[n]{a} s'$ for some $a \in L$ and $n \in \mathbf{N}$, then there is a state t' such that $t \xrightarrow[n]{a} t'$ and $(s', t') \in R$;

(ii) vice versa, for all states $s, t, t' \in S$, whenever $(s, t) \in R$ and $t \xrightarrow[n]{a} t'$ for some $a \in L$ and $n \in \mathbf{N}$, then there is a state s' such that $s \xrightarrow[n]{a} s'$ and $(s', t') \in R$;

(iii) whenever $(s, t) \in R$ and $s{\downarrow_n}$ for some $n \in \mathbf{N}$, then $t{\downarrow_n}$;

(iv) vice versa, whenever $(s, t) \in R$ and $t{\downarrow_n}$ for some $n \in \mathbf{N}$, then $s{\downarrow_n}$;

(v) whenever $(s, t) \in R$ and $s \rightsquigarrow_n$ for some $n \in \mathbf{N}$, then $t \rightsquigarrow_n$;

(vi) vice versa, whenever $(s, t) \in R$ and $t \rightsquigarrow_n$ for some $n \in \mathbf{N}$, then $s \rightsquigarrow_n$.

Two transition systems $s, t \in S$ are *timed-bisimulation equivalent* or *timed bisimilar*, notation $s \leftrightarrows_{\mathrm{t}} t$, if and only if there is a timed bisimulation relation R on S with $(s, t) \in R$.

Theorem 9.2.7 (Equivalence) Timed bisimilarity is an equivalence.

https://doi.org/10.1017/CBO9781139195003.013 Published online by Cambridge University Press

Proof The proof that bisimilarity is an equivalence can be followed exactly, in order to show that timed bisimilarity is an equivalence. □

9.3 Discrete time, relative time

This section presents theory $\mathrm{BSP}^{\mathrm{drt}}(A)$, which is a variant of $\mathrm{BSP}(A)$ with discrete, relative time. It is formally not an extension of $\mathrm{BSP}(A)$ as defined in Definition 2.2.14, because the signature of $\mathrm{BSP}(A)$ is not included in the signature of $\mathrm{BSP}^{\mathrm{drt}}(A)$. Although earlier it was argued that the 'any-time-slice' interpretation of the untimed atomic actions is more interesting, this section first presents an elementary theory with actions that take place in the *current time slice*. The signature of theory $\mathrm{BSP}^{\mathrm{drt}}(A)$ contains, besides the alternative-composition operator $+$, a *current-time-slice action-prefix* operator $\underline{a}.\_$, for any $a \in A$, and the constants *current-time-slice time stop* $\underline{0}$ and *current-time-slice termination* $\underline{1}$. The process $\underline{a}.x$ executes the action a in the current time slice and continues as the process x. The constant $\underline{0}$ expresses that time cannot progress beyond the current time slice, and no termination can take place. The current-time-slice time-stop constant is the identity element for alternative composition. The constant $\underline{1}$ expresses that time cannot progress beyond the current time slice, and that termination takes place. The current-time-slice termination constant is the identity element for sequential composition (to be added in Section 9.7). As all the atomic actions take place in the current time slice, the signature of the process theory contains the *time-prefix* operator $\sigma.\_$ to describe the passage to the next time slice explicitly. The process $\sigma.x$ passes to the next time slice and then executes x. In order to be able to distinguish between action execution and passage of time, it is assumed that $\sigma \notin A$.

As an example, the process term $\sigma.(\underline{a}.\underline{1} + \sigma.\underline{b}.\underline{1})$ specifies the process that can execute an a action in the second time slice followed by termination in that second time slice, or a b action followed by termination in the third time slice. This process term conforms to the example used in the introduction of this chapter to illustrate time determinism.

The axioms of $\mathrm{BSP}^{\mathrm{drt}}(A)$ are given in Table 9.1. They capture the properties of the operators mentioned before. Axiom DRTF, for *Discrete Relative Time Factorization*, captures the intuition that the passage of time by itself does not determine a choice, i.e., the already mentioned (weak) time determinism. Mathematically, DRTF can be paraphrased as the distribution of time prefix over alternative composition.

Example 9.3.1 (Derivation) The process terms $(\sigma.\underline{a}.\underline{1} + \sigma.\underline{b}.\underline{1}) + \sigma.\underline{0}$ and $\sigma.(\underline{a}.\underline{1} + \underline{b}.\underline{1})$ are equal in $\mathrm{BSP}^{\mathrm{drt}}(A)$, as can be seen as follows:

https://doi.org/10.1017/CBO9781139195003.013 Published online by Cambridge University Press

$\_\,\mathrm{BSP}^{\mathrm{drt}}(A)\,\_$		
constant: $\underline{0}$, $\underline{1}$;	unary: $(\underline{a}.\_)_{a\in A}$, $\sigma.\_$;	binary: $\_ + \_$;
x, y, z;		
$x + y = y + x$		A1
$(x + y) + z = x + (y + z)$		A2
$x + x = x$		A3
$x + \underline{0} = x$		A6DR
$\sigma.(x + y) = \sigma.x + \sigma.y$		DRTF

Table 9.1. The process theory $\mathrm{BSP}^{\mathrm{drt}}(A)$ (with $a \in A$).

$$\begin{aligned}\mathrm{BSP}^{\mathrm{drt}}(A) \vdash\ & (\sigma.\underline{a}.\underline{1} + \sigma.\underline{b}.\underline{1}) + \sigma.\underline{0} = \sigma.(\underline{a}.\underline{1} + \underline{b}.\underline{1}) + \sigma.\underline{0} \\ & = \sigma.((\underline{a}.\underline{1} + \underline{b}.\underline{1}) + \underline{0}) = \sigma.(\underline{a}.\underline{1} + \underline{b}.\underline{1}).\end{aligned}$$

The process theory $\mathrm{BSP}^{\mathrm{drt}}(A)$ is not an equational conservative ground-extension of the process theory $\mathrm{BSP}(A)$, since the signature of $\mathrm{BSP}(A)$ is not included in the signature of $\mathrm{BSP}^{\mathrm{drt}}(A)$. Nevertheless, there is a strong relation between these process theories. If one interprets the operators 0, 1, and $a.\_$ of $\mathrm{BSP}(A)$ as the current-time-slice variants $\underline{0}$, $\underline{1}$ and $\underline{a}.\_$ of $\mathrm{BSP}^{\mathrm{drt}}(A)$, respectively, then it turns out that all equalities of $\mathrm{BSP}(A)$ are also equalities of $\mathrm{BSP}^{\mathrm{drt}}(A)$. This kind of relation between two process theories is also called an *embedding*. The intuition of the embedding described above is the previously mentioned embedding of untimed process theory into timed process theory where all activities take place in the first time slice.

Further on, there is a need to consider closed $\mathrm{BSP}^{\mathrm{drt}}(A)$-terms in a particular form, called *basic terms*, in line with the notion of basic terms as already used in Chapter 2. In the following definition, analogous to Notation 4.6.6 (n-fold action prefix), notation $\sigma^n x$ is used for the n-fold time prefix.

Definition 9.3.2 (Basic $\mathrm{BSP}^{\mathrm{drt}}(A)$-terms) The set of *basic terms* over the signature of $\mathrm{BSP}^{\mathrm{drt}}(A)$ is defined inductively as follows:

(i) terms $\sigma^n\underline{0}$ and $\sigma^n\underline{1}$ are basic terms, for each $n \in \mathbf{N}$;
(ii) for each basic term p, action $a \in A$, and $n \in \mathbf{N}$, $\sigma^n\underline{a}.p$ is a basic term;
(iii) if p, q are basic terms, then $p + q$ is a basic term.

The following is a kind of elimination theorem: a time prefix occurring before a choice is eliminated.

Proposition 9.3.3 (Reduction to basic terms) Let p be a closed $\mathrm{BSP}^{\mathrm{drt}}(A)$-term. Then there is a basic $\mathrm{BSP}^{\mathrm{drt}}(A)$-term q such that $\mathrm{BSP}^{\mathrm{drt}}(A) \vdash p = q$.

https://doi.org/10.1017/CBO9781139195003.013 Published online by Cambridge University Press

Proof Straightforward, by using Axiom DRTF as a rewrite rule from left to right. □

Exercises

9.3.1 Complete the proof of Proposition 9.3.3 (Reduction to basic terms).

9.4 The term model

In this section, a model is constructed for $\mathrm{BSP}^{\mathrm{drt}}(A)$, by associating a timed transition system with each closed process term and then considering timed-bisimilarity equivalence classes of these timed transition systems.

Definition 9.4.1 (Term algebra) Algebra $\mathbb{P}(\mathrm{BSP}^{\mathrm{drt}}(A)) = (\mathcal{C}(\mathrm{BSP}^{\mathrm{drt}}(A)), +, (\underline{a}._-)_{a\in A}, \sigma._-, \underline{0}, \underline{1})$ is the term algebra for theory $\mathrm{BSP}^{\mathrm{drt}}(A)$.

A timed transition system is associated with each process term by means of the term deduction system presented in Table 9.2.

TDS($\mathrm{BSP}^{\mathrm{drt}}(A)$)

constant: $\underline{0}$, $\underline{1}$; unary: $(\underline{a}._-)_{a\in A}$, $\sigma._-$; binary: $\_ + \_$;

x, x', y, y';

$$x \rightsquigarrow_0 \qquad \underline{1}\downarrow_0 \qquad \underline{a}.x \ {}_0\!\xrightarrow{a}\ x$$

$$\frac{x \rightsquigarrow_n}{\sigma.x \rightsquigarrow_{n+1}} \qquad \frac{x\downarrow_n}{\sigma.x\downarrow_{n+1}} \qquad \frac{x \ {}_n\!\xrightarrow{a}\ x'}{\sigma.x \ {}_{n+1}\!\xrightarrow{a}\ x'}$$

$$\frac{x \rightsquigarrow_n}{x + y \rightsquigarrow_n} \qquad \frac{x\downarrow_n}{(x + y)\downarrow_n} \qquad \frac{y\downarrow_n}{(x + y)\downarrow_n}$$

$$\frac{y \rightsquigarrow_n}{x + y \rightsquigarrow_n} \qquad \frac{x \ {}_n\!\xrightarrow{a}\ x'}{x + y \ {}_n\!\xrightarrow{a}\ x'} \qquad \frac{y \ {}_n\!\xrightarrow{a}\ y'}{x + y \ {}_n\!\xrightarrow{a}\ y'}$$

Table 9.2. Term deduction system for $\mathrm{BSP}^{\mathrm{drt}}(A)$ (with $a \in A, n \in \mathbf{N}$).

Example 9.4.2 (Timed transition system of a $\mathrm{BSP}^{\mathrm{drt}}(A)$-term) The timed transition system of Figure 9.2 is the transition system associated to the process term $\sigma.(\underline{a}.\sigma.\underline{b}.\sigma.\underline{1} + \sigma.\underline{0})$.

https://doi.org/10.1017/CBO9781139195003.013 Published online by Cambridge University Press

The first axiom of the term deduction system of Table 9.2 says that every process in the present theory satisfies the predicate $\rightsquigarrow_0$, i.e., every process allows zero time ticks to occur. In some theories with timing, processes occur that denote an inconsistency, e.g., a timing inconsistency in an absolute-time theory. Such inconsistencies can be operationally characterized by the absence of the predicate $\rightsquigarrow_0$.

As already mentioned, the predicates $\rightsquigarrow_n$ can in general be used to specify timing behavior independent of actions and termination. In the present context, they are needed to distinguish processes that only differ in their timing behavior. To give an example, the processes specified by terms $\underline{0}$ and $\sigma.\underline{0}$ are only distinguished by the predicate $\rightsquigarrow_1$.

With the given term deduction system, the construction of a term model goes along the lines as before.

Theorem 9.4.3 (Congruence) Timed bisimilarity is a congruence on the algebra $\mathbb{P}(\mathrm{BSP}^{\mathrm{drt}}(A))$.

Proof The results presented in Chapter 3 do not apply directly to prove the desired congruence result, because the notion of transition systems used in the current chapter is different. However, (Baeten & Verhoef, 1995) shows that the meta-theory of Chapter 3 generalizes to transition-system spaces with multiple types of transitions and multiple predicates on states. The timed transition systems of the present setting fit into the framework of (Baeten & Verhoef, 1995), and the congruence result follows from the path format, as defined in (Baeten & Verhoef, 1995) for the general setting, of the deduction rules of Table 9.2. □

Definition 9.4.4 (Term model of $\mathrm{BSP}^{\mathrm{drt}}(A)$) The term model of process theory $\mathrm{BSP}^{\mathrm{drt}}(A)$ is the quotient algebra $\mathbb{P}(\mathrm{BSP}^{\mathrm{drt}}(A))_{/\leftrightarrows_t}$.

Theorem 9.4.5 (Soundness) Theory $\mathrm{BSP}^{\mathrm{drt}}(A)$ is a sound axiomatization of the algebra $\mathbb{P}(\mathrm{BSP}^{\mathrm{drt}}(A))_{/\leftrightarrows_t}$, i.e., $\mathbb{P}(\mathrm{BSP}^{\mathrm{drt}}(A))_{/\leftrightarrows_t} \models \mathrm{BSP}^{\mathrm{drt}}(A)$.

Proof See Exercise 9.4.1. □

Next, it is shown that $\mathrm{BSP}^{\mathrm{drt}}(A)$ is a ground-complete axiomatization of the term model $\mathbb{P}(\mathrm{BSP}^{\mathrm{drt}}(A))_{/\leftrightarrows_t}$. Thereto, the following two lemmas are introduced. Note the correspondence with the Lemmas 4.4.10 (Towards completeness) and 4.4.11 used in obtaining the ground-completeness of $\mathrm{BSP}(A)$ (Theorem 4.4.12).

https://doi.org/10.1017/CBO9781139195003.013 Published online by Cambridge University Press

Lemma 9.4.6 (Towards completeness) For arbitrary closed $\mathrm{BSP}^{\mathrm{drt}}(A)$-terms p and p' and arbitrary $a \in A, n \in \mathbf{N}$:

(i) if $p \rightsquigarrow_n$, then $\mathrm{BSP}^{\mathrm{drt}}(A) \vdash p = \sigma^n(\underline{0}) + p$;
(ii) if $p{\downarrow_n}$, then $\mathrm{BSP}^{\mathrm{drt}}(A) \vdash p = \sigma^n(\underline{1}) + p$;
(iii) if $p \xrightarrow[n]{a} p'$, then $\mathrm{BSP}^{\mathrm{drt}}(A) \vdash p = \sigma^n(\underline{a}.p') + p$.

Proof These three properties are proven by induction on n. In the inductive step, induction on the structure of closed $\mathrm{BSP}^{\mathrm{drt}}(A)$-term p is used. Details are left as an exercise to the reader (Exercise 9.4.2). □

Lemma 9.4.7 Let p, q, and r be closed $\mathrm{BSP}^{\mathrm{drt}}(A)$-terms. If $(p+q)+r \leftrightarrows_{\mathrm{t}} r$, then $p + r \leftrightarrows_{\mathrm{t}} r$ and $q + r \leftrightarrows_{\mathrm{t}} r$.

Proof See Exercise 9.4.3. □

Theorem 9.4.8 (Ground-completeness) Equational theory $\mathrm{BSP}^{\mathrm{drt}}(A)$ is a ground-complete axiomatization of the term model $\mathbb{P}(\mathrm{BSP}^{\mathrm{drt}}(A))_{/\leftrightarrows_{\mathrm{t}}}$, i.e., for any closed $\mathrm{BSP}^{\mathrm{drt}}(A)$-terms p and q, $\mathbb{P}(\mathrm{BSP}^{\mathrm{drt}}(A))_{/\leftrightarrows_{\mathrm{t}}} \models p = q$ implies $\mathrm{BSP}^{\mathrm{drt}}(A) \vdash p = q$.

Proof Due to Proposition 9.3.3 (Reduction to basic terms) and the soundness of the axioms of $\mathrm{BSP}^{\mathrm{drt}}(A)$ (Theorem 9.4.5), it suffices to prove this theorem for basic $\mathrm{BSP}^{\mathrm{drt}}(A)$-terms.
Let p and q be basic terms. Suppose that $\mathbb{P}(\mathrm{BSP}^{\mathrm{drt}}(A))_{/\leftrightarrows_{\mathrm{t}}} \models p = q$, i.e., $p \leftrightarrows_{\mathrm{t}} q$. It must be shown that $\mathrm{BSP}^{\mathrm{drt}}(A) \vdash p = q$. Since bisimilarity is a congruence on $\mathbb{P}(\mathrm{BSP}^{\mathrm{drt}}(A))$ (Theorem 9.4.3) and $\mathrm{BSP}^{\mathrm{drt}}(A)$ is sound for $\mathbb{P}(\mathrm{BSP}^{\mathrm{drt}}(A))_{/\leftrightarrows_{\mathrm{t}}}$ (Theorem 9.4.5), in line with the reasoning in the proof of Theorem 4.3.10 (Ground-completeness of $\mathrm{MPT}(A)$), it suffices to prove that, for all basic $\mathrm{BSP}^{\mathrm{drt}}(A)$-terms p and q,

$$p + q \leftrightarrows_{\mathrm{t}} q \quad \text{implies} \quad \mathrm{BSP}^{\mathrm{drt}}(A) \vdash p + q = q \tag{9.4.1}$$

and

$$p \leftrightarrows_{\mathrm{t}} p + q \quad \text{implies} \quad \mathrm{BSP}^{\mathrm{drt}}(A) \vdash p = p + q. \tag{9.4.2}$$

Property (9.4.1) is proven by induction on the total number of symbols in terms p and q. The proof of property (9.4.2) is similar and therefore omitted. Assume $p + q \leftrightarrows_{\mathrm{t}} q$ for basic terms p and q. The base case of the induction corresponds to the case that the total number of symbols in p and q equals two, namely when p and q are both equal to $\underline{0}$ or $\underline{1}$. In the case that $p \equiv \underline{0}$, using Axiom A6DR, it trivially follows that $\mathrm{BSP}^{\mathrm{drt}}(A) \vdash p + q = q$. In

https://doi.org/10.1017/CBO9781139195003.013 Published online by Cambridge University Press

the case that $p \equiv \underline{1}$, $p{\downarrow_0}$ and thus $(p+q){\downarrow_0}$. As $p + q \leftrightarrows_t q$, also $q{\downarrow_0}$. By Lemma 9.4.6 (Towards completeness), $\mathrm{BSP}^{\mathrm{drt}}(A) \vdash q = \underline{1} + q$. Hence, $\mathrm{BSP}^{\mathrm{drt}}(A) \vdash p + q = \underline{1} + q = q$. The proof of the inductive step consists of a case analysis based on the structure of term p.

(i) Assume $p \equiv \sigma^n\underline{0}$ for some $n \in \mathbf{N}$. By Table 9.2, it follows that $p \rightsquigarrow_n$, and so also $p + q \rightsquigarrow_n$. As $p + q \leftrightarrows_t q$, also $q \rightsquigarrow_n$. By Lemma 9.4.6 (Towards completeness), $\mathrm{BSP}^{\mathrm{drt}}(A) \vdash q = \sigma^n\underline{0} + q$. Then, $\mathrm{BSP}^{\mathrm{drt}}(A) \vdash p + q = \sigma^n\underline{0} + q = q$.

(ii) Assume $p \equiv \sigma^n\underline{1}$ for some $n \in \mathbf{N}$. It follows from Table 9.2 that $p{\downarrow_n}$, and therefore also $(p+q){\downarrow_n}$. As $p + q \leftrightarrows_t q$, also $q{\downarrow_n}$. By Lemma 9.4.6 (Towards completeness), $\mathrm{BSP}^{\mathrm{drt}}(A) \vdash q = \sigma^n\underline{1} + q$. Thus, $\mathrm{BSP}^{\mathrm{drt}}(A) \vdash p + q = \sigma^n\underline{1} + q = q$.

(iii) Assume $p \equiv \sigma^n\underline{a}.p'$ for some $a \in A, n \in \mathbf{N}$ and basic term p'. Then, $p \xrightarrow{a}_n p'$ and thus $p + q \xrightarrow{a}_n p'$. As $p + q \leftrightarrows_t q$, also $q \xrightarrow{a}_n q'$ for some basic term q' such that $p' \leftrightarrows_t q'$. By Lemma 9.4.6 (Towards completeness), $\mathrm{BSP}^{\mathrm{drt}}(A) \vdash q = \sigma^n\underline{a}.q' + q$. From $p' \leftrightarrows_t q'$, following the reasoning in the first part of the proof of Theorem 4.3.10 (Ground-completeness of MPT(A)), it follows that $p'+q' \leftrightarrows_t q'$ and $q' + p' \leftrightarrows_t p'$ and, hence, by induction, $\mathrm{BSP}^{\mathrm{drt}}(A) \vdash p'+q' = q'$ and $\mathrm{BSP}^{\mathrm{drt}}(A) \vdash q'+p' = p'$. Combining these last two results gives $\mathrm{BSP}^{\mathrm{drt}}(A) \vdash p' = q' + p' = p' + q' = q'$. Finally, $\mathrm{BSP}^{\mathrm{drt}}(A) \vdash p + q = \sigma^n\underline{a}.p' + q = \sigma^n\underline{a}.q' + q = q$.

(iv) Assume $p \equiv p_1 + p_2$ for basic terms p_1 and p_2. As $(p_1+p_2)+q \leftrightarrows_t q$, by Lemma 9.4.7, $p_1 + q \leftrightarrows_t q$ and $p_2 + q \leftrightarrows_t q$. Thus, by induction, $\mathrm{BSP}^{\mathrm{drt}}(A) \vdash p_1 + q = q$ and $\mathrm{BSP}^{\mathrm{drt}}(A) \vdash p_2 + q = q$. Combining these results gives $\mathrm{BSP}^{\mathrm{drt}}(A) \vdash p+q = (p_1 + p_2)+q = p_1 + (p_2 + q) = p_1 + q = q$, which completes the proof. □

Exercises

9.4.1 Prove Theorem 9.4.5 (Soundness of $\mathrm{BSP}^{\mathrm{drt}}(A)$).

9.4.2 Prove Lemma 9.4.6 (Towards completeness).

9.4.3 Prove Lemma 9.4.7.

9.5 Time iteration and delayable actions

In this section, the timed process algebra $\mathrm{BSP}^{\mathrm{drt}}(A)$ from the previous two sections is extended with the constants *any-time-slice deadlock* 0 and *any-time-slice termination* 1, the *any-time-slice action-prefix* operators $a._-$ (for $a \in A$)

https://doi.org/10.1017/CBO9781139195003.013 Published online by Cambridge University Press

and the auxiliary *time-iteration prefix* operator $\sigma^*\_$ to obtain the process theory $\mathrm{BSP}^{\mathrm{drt}^*}(A)$. The constants 0 and 1 are delayable versions of $\underline{0}$ and $\underline{1}$ which can take effect in any time slice (present or future). Similarly, $a.p$ denotes the execution of a in an arbitrary time slice followed by execution of p, and $\sigma^* p$ denotes that the execution of p can be started in any time slice. Note that the intuitions of 0, 1 and $a.\_$ are in line with the 'any-time-slice' interpretation of the untimed constants and action-prefix operators.

$\mathrm{BSP}^{\mathrm{drt}^*}(A)$

$\mathrm{BSP}^{\mathrm{drt}}(A)$

constant: 0, 1; unary: $(a.\_)_{a\in A}$, $\sigma^*\_$;

x, y;

$0 = \sigma^*\underline{0}$	DD	$\sigma^* x = x + \sigma.\sigma^* x$	ATS
$1 = \sigma^*\underline{1}$	DT	$\sigma^* x + \sigma^* y = \sigma^*(x + y)$	DRTIF
$a.x = \sigma^*\underline{a}.x$	DA	$\sigma^*\sigma.x = \sigma.\sigma^* x$	DRTA
		$\sigma^*\sigma^* x = \sigma^* x$	TITI

Table 9.3. The process theory $\mathrm{BSP}^{\mathrm{drt}^*}(A)$ (with $a \in A$).

The axioms of theory $\mathrm{BSP}^{\mathrm{drt}^*}(A)$ are given in Table 9.3. Axioms DD (Delayable Deadlock), DT (Delayable Termination) and DA (Delayable Actions) define the any-time-slice constants and action-prefix operators in terms of their current-time-slice counterparts and time iteration. Axiom DA illustrates that the any-time-slice action prefix can be interpreted as a *delayable action*. Axiom ATS (Any Time Slice) recursively defines time iteration. Axiom DRTIF (Discrete Relative Time Iteration Factorization) expresses that time factorization, i.e., the equational equivalent of (weak) time determinism, also applies to time iteration. Axiom DRTA (Discrete Relative Time Axiom) explains that also for time iteration time is measured relative to the previous action execution. Axiom TITI (Time Iteration Time Iteration) says that two consecutive time iterations are equivalent to only one time iteration. As a consequence, any number of consecutive time iterations is considered to be equivalent to a single one.

It is not the case that the newly introduced operators can all be eliminated. Nevertheless, the newly introduced syntax has some redundancy in the sense that either the time-iteration operator or the other newly introduced operators can be eliminated from closed terms. Definition 9.3.2 (Basic $\mathrm{BSP}^{\mathrm{drt}}(A)$-terms) is extended in order to incorporate the delayable constants and actions. Note that it uses the n-fold time prefix notation introduced just before Definition 9.3.2.

https://doi.org/10.1017/CBO9781139195003.013 Published online by Cambridge University Press

Definition 9.5.1 (Basic $\mathrm{BSP}^{\mathrm{drt}^*}(A)$-terms) The set of *basic terms* over the signature of theory $\mathrm{BSP}^{\mathrm{drt}^*}(A)$ is defined inductively as follows:

(i) terms $\sigma^n\underline{0}$, $\sigma^n 0$, $\sigma^n\underline{1}$, and $\sigma^n 1$ are basic terms, for each $n \in \mathbf{N}$;
(ii) for each basic term p, action $a \in A$, and $n \in \mathbf{N}$, $\sigma^n\underline{a}.p$ and $\sigma^n a.p$ are basic terms;
(iii) if p, q are basic terms, then $p + q$ is a basic term.

Again, a closed term can be reduced to a basic term, thereby eliminating the time-iteration operator.

Proposition 9.5.2 (Reduction to basic terms) Let p be a closed $\mathrm{BSP}^{\mathrm{drt}^*}(A)$-term. There is a basic $\mathrm{BSP}^{\mathrm{drt}^*}(A)$-term q such that $\mathrm{BSP}^{\mathrm{drt}^*}(A) \vdash p = q$.

Proof See Exercise 9.5.2. □

Theorem 9.5.3 (Conservative ground-extension) Theory $\mathrm{BSP}^{\mathrm{drt}^*}(A)$ is a conservative ground-extension of theory $\mathrm{BSP}^{\mathrm{drt}}(A)$.

Proof The theorem follows from meta-results in the style of the results of Chapter 3, in particular Theorem 3.2.21 (Conservativity), for the generalized operational framework with transition systems with multiple types of transitions and/or state predicates as treated in (Baeten & Verhoef, 1995). The proof is based on the term model given below (although the result is model independent). □

A model is constructed for theory $\mathrm{BSP}^{\mathrm{drt}^*}(A)$ by associating a timed transition system with each closed term and then considering timed-bisimilarity equivalence classes of these timed transition systems. The term algebra for theory $\mathrm{BSP}^{\mathrm{drt}^*}(A)$ is the algebra $\mathbb{P}(\mathrm{BSP}^{\mathrm{drt}^*}(A)) = (\mathcal{C}(\mathrm{BSP}^{\mathrm{drt}^*}(A)), +, (a._-)_{a\in A}, (\underline{a}._-)_{a\in A}, \sigma._-, 0, \underline{0}, 1, \underline{1})$. The term deduction system of Table 9.4 gives a timed transition system for each term in this algebra.

Proposition 9.5.4 (Congruence) Timed bisimilarity is a congruence on algebra $\mathbb{P}(\mathrm{BSP}^{\mathrm{drt}^*}(A))$.

Proof As before, it follows from the format of the deduction rules; see also the proof of Theorem 9.4.3. □

The term model of $\mathrm{BSP}^{\mathrm{drt}^*}(A)$ is the quotient algebra $\mathbb{P}(\mathrm{BSP}^{\mathrm{drt}^*}(A))_{/\leftrightarrows_t}$.

Theorem 9.5.5 (Soundness) Theory $\mathrm{BSP}^{\mathrm{drt}^*}(A)$ is a sound axiomatization of the algebra $\mathbb{P}(\mathrm{BSP}^{\mathrm{drt}^*}(A))_{/\leftrightarrows_t}$, i.e., $\mathbb{P}(\mathrm{BSP}^{\mathrm{drt}^*}(A))_{/\leftrightarrows_t} \models \mathrm{BSP}^{\mathrm{drt}^*}(A)$.

$TDS(\mathrm{BSP}^{\mathrm{drt}^*}(A))$
$TDS(\mathrm{BSP}^{\mathrm{drt}}(A))$

constant: $0, 1$; unary: $(a._-)_{a\in A}, \sigma^*_-$;

x, x';

$$1{\downarrow_n} \qquad 1 \rightsquigarrow_n \qquad 0 \rightsquigarrow_n$$

$$a.x \ {}_n\!\xrightarrow{a}\ x \qquad a.x \rightsquigarrow_n$$

$$\sigma^*x \rightsquigarrow_n \qquad \frac{x{\downarrow_n}}{\sigma^*x{\downarrow_{n+m}}} \qquad \frac{x \ {}_n\!\xrightarrow{a}\ x'}{\sigma^*x \ {}_{n+m}\!\xrightarrow{a}\ x'}$$

Table 9.4. Term deduction system for $\mathrm{BSP}^{\mathrm{drt}^*}(A)$ ($a \in A, n, m \in \mathbf{N}$).

Proof See Exercise 9.5.3. □

Also ground-completeness can be shown as before. The following lemma is in line with the lemmas for earlier ground-completeness proofs.

Lemma 9.5.6 (Towards completeness) For any closed $\mathrm{BSP}^{\mathrm{drt}^*}(A)$-terms p and p' and arbitrary $a \in A, n \in \mathbf{N}$:

(i) if $p \rightsquigarrow_n$, then $\mathrm{BSP}^{\mathrm{drt}^*}(A) \vdash p = \sigma^n\underline{0} + p$;
(ii) if for all $m \geq n$, $p \rightsquigarrow_m$, then $\mathrm{BSP}^{\mathrm{drt}^*}(A) \vdash p = \sigma^n 0 + p$;
(iii) if $p{\downarrow_n}$, then $\mathrm{BSP}^{\mathrm{drt}^*}(A) \vdash p = \sigma^n\underline{1} + p$;
(iv) if for all $m \geq n$, $p{\downarrow_m}$, then $\mathrm{BSP}^{\mathrm{drt}^*}(A) \vdash p = \sigma^n 1 + p$;
(v) if $p \ {}_n\!\xrightarrow{a}\ p'$, then $\mathrm{BSP}^{\mathrm{drt}^*}(A) \vdash p = \sigma^n\underline{a}.p' + p$;
(vi) if for all $m \geq n$, $p \ {}_m\!\xrightarrow{a}\ p'$, then $\mathrm{BSP}^{\mathrm{drt}^*}(A) \vdash p = \sigma^n a.p' + p$.

Proof As in the proof of Lemma 9.4.6, these properties are proven via induction on n, using structural induction in the inductive step. Based on Proposition 9.5.2 (Reduction to basic terms) and Theorem 9.5.5 (Soundness of $\mathrm{BSP}^{\mathrm{drt}^*}(A)$), it is sufficient to prove the properties for basic terms. The proofs need the following fact: if p is a basic $\mathrm{BSP}^{\mathrm{drt}^*}(A)$-term and $p \ {}_n\!\xrightarrow{a}\ p'$ for some $a \in A, n \in \mathbf{N}$, then also p' is a basic $\mathrm{BSP}^{\mathrm{drt}^*}(A)$-term. This can easily be established by induction on the depth of the derivation of the transition. □

Lemma 9.5.7 Let p, q, and r be closed $\mathrm{BSP}^{\mathrm{drt}^*}(A)$-terms. If $(p+q)+r \leftrightarrows_{\mathrm{t}} r$, then $p + r \leftrightarrows_{\mathrm{t}} r$ and $q + r \leftrightarrows_{\mathrm{t}} r$.

Proof See Exercise 9.5.4. □

https://doi.org/10.1017/CBO9781139195003.013 Published online by Cambridge University Press

Theorem 9.5.8 (Ground-completeness) Theory $\mathrm{BSP}^{\mathrm{drt}^*}(A)$ is a ground-complete axiomatization of the term model $\mathbb{P}(\mathrm{BSP}^{\mathrm{drt}^*}(A))_{/\leftrightarrows_{\mathrm{t}}}$, i.e., for any closed $\mathrm{BSP}^{\mathrm{drt}^*}(A)$-terms p and q, $\mathbb{P}(\mathrm{BSP}^{\mathrm{drt}^*}(A))_{/\leftrightarrows_{\mathrm{t}}} \models p = q$ implies $\mathrm{BSP}^{\mathrm{drt}^*}(A) \vdash p = q$.

Proof Due to Proposition 9.5.2 (Reduction to basic terms) and the soundness of the axioms of $\mathrm{BSP}^{\mathrm{drt}^*}(A)$ (Theorem 9.5.5), it suffices to prove this theorem for basic terms. Let p and q be basic $\mathrm{BSP}^{\mathrm{drt}^*}(A)$-terms. Suppose that $\mathbb{P}(\mathrm{BSP}^{\mathrm{drt}^*}(A))_{/\leftrightarrows_{\mathrm{t}}} \models p = q$, i.e., $p \leftrightarrows_{\mathrm{t}} q$. It must be shown that $\mathrm{BSP}^{\mathrm{drt}^*}(A) \vdash p = q$. Since bisimilarity is a congruence on $\mathbb{P}(\mathrm{BSP}^{\mathrm{drt}^*}(A))$ (Proposition 9.5.4) and $\mathrm{BSP}^{\mathrm{drt}^*}(A)$ is sound for $\mathbb{P}(\mathrm{BSP}^{\mathrm{drt}^*}(A))_{/\leftrightarrows_{\mathrm{t}}}$ (Theorem 9.5.5), it suffices to prove that, for all basic terms p and q,

$$p + q \leftrightarrows_{\mathrm{t}} q \quad \text{implies} \quad \mathrm{BSP}^{\mathrm{drt}^*}(A) \vdash p + q = q \tag{9.5.1}$$

and

$$p \leftrightarrows_{\mathrm{t}} p + q \quad \text{implies} \quad \mathrm{BSP}^{\mathrm{drt}^*}(A) \vdash p = p + q. \tag{9.5.2}$$

Property (9.5.1) is proven by induction on the total number of symbols in basic terms p and q. The proof of property (9.5.2) is similar and therefore omitted. Assume $p + q \leftrightarrows_{\mathrm{t}} q$ for some basic terms p and q. The base case of the induction corresponds to the case that the total number of symbols in p and q equals two, namely when p and q are both equal to $\underline{0}$, 0, $\underline{1}$, or 1.

(i) Assume $p \equiv \underline{0}$. Using Axioms A6DR and A1, it trivially follows that $\mathrm{BSP}^{\mathrm{drt}^*}(A) \vdash p + q = \underline{0} + q = q + \underline{0} = q$.

(ii) Assume $p \equiv 0$. By Table 9.4, $p \rightsquigarrow_n$ for all $n \in \mathbf{N}$. By Table 9.2 also $p + q \rightsquigarrow_n$ for all $n \in \mathbf{N}$. Therefore, as $p + q \leftrightarrows_{\mathrm{t}} q$, also $q \rightsquigarrow_n$ for all $n \in \mathbf{N}$. Following Lemma 9.5.6 (ii), $\mathrm{BSP}^{\mathrm{drt}^*}(A) \vdash q = 0 + q$. Then, $\mathrm{BSP}^{\mathrm{drt}^*}(A) \vdash p + q = 0 + q = q$.

(iii) Assume $p \equiv \underline{1}$. Then, $p{\downarrow_0}$ and thus $(p + q){\downarrow_0}$. As $p + q \leftrightarrows_{\mathrm{t}} q$, also $q{\downarrow_0}$. By Lemma 9.5.6 (iii), $\mathrm{BSP}^{\mathrm{drt}^*}(A) \vdash q = \underline{1} + q$. Hence, $\mathrm{BSP}^{\mathrm{drt}^*}(A) \vdash p + q = \underline{1} + q = q$.

(iv) Assume $p \equiv 1$. By Table 9.4, $p{\downarrow_n}$ for all $n \in \mathbf{N}$. By Table 9.2 also $(p + q){\downarrow_n}$ for all $n \in \mathbf{N}$. Therefore, as $p + q \leftrightarrows_{\mathrm{t}} q$, also $q{\downarrow_n}$ for all $n \in \mathbf{N}$. Following Lemma 9.5.6 (iv), $\mathrm{BSP}^{\mathrm{drt}^*}(A) \vdash q = 1 + q$. Then, $\mathrm{BSP}^{\mathrm{drt}^*}(A) \vdash p + q = 1 + q = q$.

The proof of the inductive step consists of a case analysis based on the structure of term p. Using Lemmas 9.5.6 and 9.5.7, this proceeds as in the proof of Theorem 9.4.8 (Ground-completeness $\mathrm{BSP}^{\mathrm{drt}}(A)$). □

https://doi.org/10.1017/CBO9781139195003.013 Published online by Cambridge University Press

Exercises

9.5.1 Prove the following for all BSP$^{\mathrm{drt}^*}$(A)-terms x and y, and all $a \in A$:

(a) BSP$^{\mathrm{drt}^*}$(A) $\vdash a.x + 0 = a.x$;
(b) BSP$^{\mathrm{drt}^*}$(A) $\vdash \sigma.x + \sigma^* y = y + \sigma.(x + \sigma^* y)$;
(c) BSP$^{\mathrm{drt}^*}$(A) $\vdash a.x + \underline{a}.x = a.x$;
(d) BSP$^{\mathrm{drt}^*}$(A) $\vdash \sigma.x + \sigma^* x = \sigma^* x$.

9.5.2 Prove Proposition 9.5.2 (Reduction to basic terms).
9.5.3 Prove Theorem 9.5.5 (Soundness).
9.5.4 Prove Lemmas 9.5.6 (Towards completeness) and 9.5.7.

9.6 The relation between BSP(A) and BSP$^{\mathrm{drt}^*}$(A)

This section discusses the relation between the untimed process theory BSP(A) and the timed theory BSP$^{\mathrm{drt}^*}$(A) in some detail. Usually, when extending a theory, one adds constants and/or operators to the signature, extends the set of axioms, and extends the original deduction system in such a way that with respect to the old signature nothing changes.

Comparing the signatures of BSP(A) and BSP$^{\mathrm{drt}^*}$(A), it can be observed that indeed the latter is an extension of the former. The extension consists of the constants $\underline{0}$ and $\underline{1}$, current-time-slice action-prefix operators $\underline{a}._-$, the time-prefix operator $\sigma._-$, and the time-iteration prefix operator $\sigma^*{}_-$. However, theory BSP$^{\mathrm{drt}^*}$(A) is not an extension of BSP(A) as defined in Definition 2.2.14, because Axiom A6 of BSP(A) is not included in BSP$^{\mathrm{drt}^*}$(A). It can be shown though that theory BSP$^{\mathrm{drt}^*}$(A) is a *ground*-extension, as defined in Definition 2.2.18, see Exercise 9.6.1. Furthermore, as explained below, it turns out that BSP$^{\mathrm{drt}^*}$(A) is a *conservative* ground-extension of BSP(A).

In the operational framework, instead of the transitions and terminations present in the untimed setting, the timed setting has natural numbers indicating the time slice as extra parameters, and the possible delays as extra predicates. Nevertheless, any two closed BSP(A)-terms that are timed bisimilar in the timed setting are also bisimilar in the untimed setting, and vice versa.

Proposition 9.6.1 (Timed vs. strong bisimilarity) For closed BSP(A)-terms p and q, $p \leftrightarrows_{\mathrm{t}} q$ if and only if $p \leftrightarrows q$.

Proof Suppose that $p \leftrightarrows_{\mathrm{t}} q$. Assume that this timed bisimilarity is witnessed by the relation R on closed BSP$^{\mathrm{drt}^*}$(A)-terms, i.e., R is a timed bisimulation and $(p, q) \in R$. Notice that if $p \ {}_n\!\xrightarrow{a}\ p'$, then p' is a subterm of p, so if p is a BSP(A)-term, then also p' is a BSP(A)-term. If follows that the

https://doi.org/10.1017/CBO9781139195003.013 Published online by Cambridge University Press

transition system of a BSP(A)-term in the timed model has the same states as in the untimed model. Moreover, $p \xrightarrow{a} p'$ holds in the untimed model exactly when $p \ {}_n\!\xrightarrow{a} p'$ holds in the timed model (for all $n \in \mathbf{N}$), and, similarly, $p\!\downarrow$ holds in the untimed model exactly when $p\!\downarrow_n$ holds in the timed model (for all $n \in \mathbf{N}$). Thus, the relation R is also a bisimulation relation on the term algebra of BSP(A), showing that $p \leftrightarrows q$. The other implication follows from a similar reasoning. □

Using among others this proposition, it can be shown that $\mathrm{BSP}^{\mathrm{drt}^*}(A)$ is a conservative ground-extension of BSP(A).

Theorem 9.6.2 (Conservative ground-extension) Theory $\mathrm{BSP}^{\mathrm{drt}^*}(A)$ is a conservative ground-extension of theory BSP(A).

Proof According to Definitions 2.2.18 (Ground-extension) and 2.2.19 (Conservative ground-extension), it needs to be shown that, for all closed BSP(A)-terms p and q, $\mathrm{BSP}(A) \vdash p = q$ if and only if $\mathrm{BSP}^{\mathrm{drt}^*}(A) \vdash p = q$. The implication from left to right, showing that $\mathrm{BSP}^{\mathrm{drt}^*}(A)$ is a ground-extension of BSP(A), follows from the observation that all axioms of BSP(A) with the exception of Axiom A6 are also axioms of $\mathrm{BSP}^{\mathrm{drt}^*}(A)$ and the first part of Exercise 9.6.1. The other implication, showing conservativity, can be shown as follows. If $\mathrm{BSP}^{\mathrm{drt}^*}(A) \vdash p = q$ for closed BSP(A)-terms p and q, it follows from the soundness of $\mathrm{BSP}^{\mathrm{drt}^*}(A)$ that $p \leftrightarrows_{\mathrm{t}} q$. Proposition 9.6.1 (Timed vs. strong bisimilarity) shows that then also $p \leftrightarrows q$. Since BSP(A) is ground-complete, Theorem 4.4.12, it follows that $\mathrm{BSP}(A) \vdash p = q$, completing the proof. As an alternative, it is possible to prove the result via meta-results in the style of Chapter 3. The interested reader is referred to (Baeten *et al.*, 2005). □

The process algebra ACP (Bergstra & Klop, 1984a; Baeten & Weijland, 1990) has action constants instead of action prefixing. At this point, a drawback of the ACP approach can be appreciated. For, an action constant a in ACP satisfies the law $a \cdot 1 = a$, and it has the operational rule $a \xrightarrow{a} 1$. This means, interpreted in a timed theory, that after a number of time steps, a is executed, *necessarily* followed by the option to execute any number of time steps followed by termination. This means that processes like $a.\underline{1}$ or $a.\underline{0}$ cannot be expressed in a conservative timed extension of ACP. If, as is common in ACP process algebra, the process 1 does not exist, this problem nevertheless resurfaces when silent actions are added, for then the law $a \cdot \tau = a$ holds, leading to $a \xrightarrow{a} \tau$, and again in a timed setting necessarily any number of time steps are

allowed to be executed after execution of a. For a more elaborate discussion on this issue, the interested reader is referred to (Baeten & Reniers, 2007).

Exercises

9.6.1 Prove that $\mathrm{BSP}^{\mathrm{drt}^*}(A) \vdash p + 0 = p$ for closed $\mathrm{BSP}(A)$-terms p. Also, give a counterexample for $\mathrm{BSP}^{\mathrm{drt}^*}(A) \vdash p + 0 = p$ for closed $\mathrm{BSP}^{\mathrm{drt}^*}(A)$-terms p.

9.7 The process theory $\mathrm{TCP}^{\mathrm{drt}^*}(A, \gamma)$

In this section, the process theory $\mathrm{BSP}^{\mathrm{drt}^*}(A)$ is extended to the process theory $\mathrm{TCP}^{\mathrm{drt}^*}(A, \gamma)$. This extension is obtained by extending the signature of $\mathrm{BSP}^{\mathrm{drt}^*}(A)$ with the *sequential-composition* operator $\_ \cdot \_$, the *current-time-slice timeout* operator ν, the *encapsulation* operators ∂_H (for $H \subseteq A$), and the *parallel-composition* operators $\_ \parallel \_$, $\_ \between \_$, and $\_ \mid \_$.

The axioms of the process theory are given in Table 9.5. Sequential composition is as before, but here the role of identity that was played by 1 in the untimed theory, is taken over by the current-time-slice termination constant $\underline{1}$ (see Axioms A8DR and A9DR). In this setting, it can be derived that $1 \cdot x = \sigma^* x$ instead. The sequential-composition operator has two left-zero elements: both undelayable inaction (see Axiom A7DR) and delayable inaction act as such. The axiom $0 \cdot x = 0$ (A7) has disappeared since it is derivable from the remaining axioms. The axiom $a.x \cdot y = a.(x \cdot y)$ (A10) from the untimed theory is now also derivable. Axioms A10DRb and A10DRc express that the passage of time is measured relative to the previous action and thus has no consequences for the future actions: the timing of y is relative to the last action of x, regardless of the time-prefix or time-iteration operator.

The current-time-slice timeout operator, with Axioms RTO1–5 (from Relative TimeOut), disallows all initial passage of time. It extracts the part of the behavior that executes an action or performs termination in the current time slice. Notice that the equation $\nu(\sigma^* x) = \nu(x)$ can be derived from the axioms given. The encapsulation operator is as defined before: encapsulation disallows the actions that occur in the set H and allows all other behavior including passage of time.

Before continuing with the other operators and axioms in the theory, the following proposition summarizes some simple identities concerning sequential composition, encapsulation, and the current-time-slice timeout operator that can be derived from the theory.

https://doi.org/10.1017/CBO9781139195003.013 Published online by Cambridge University Press

$\mathrm{TCP}^{\mathrm{drt}*}(A,\gamma)$

$\mathrm{BSP}^{\mathrm{drt}*}(A)$;

unary: ν, $(\partial_H)_{H\subseteq A}$; binary: $\_\cdot\_$, $\_\parallel\_$, $\_\between\_$, $\_\mid\_$;

x, y, z;

$(x+y)\cdot z = x\cdot z + y\cdot z$	A4	$\nu(\underline{1}) = \underline{1}$	RTO1
$(x\cdot y)\cdot z = x\cdot(y\cdot z)$	A5	$\nu(\underline{0}) = \underline{0}$	RTO2
$\underline{0}\cdot x = \underline{0}$	A7DR	$\nu(\underline{a}.x) = \underline{a}.x$	RTO3
$x\cdot\underline{1} = x$	A8DR	$\nu(x+y) = \nu(x)+\nu(y)$	RTO4
$\underline{1}\cdot x = x$	A9DR	$\nu(\sigma.x) = \underline{0}$	RTO5
$\underline{a}.x\cdot y = \underline{a}.(x\cdot y)$	A10DRa		
$(\sigma.x)\cdot y = \sigma.(x\cdot y)$	A10DRb		
$\sigma^*x\cdot y = \sigma^*(x\cdot y)$	A10DRc		

$\partial_H(\underline{1}) = \underline{1}$		D1DR	$\partial_H(x+y) = \partial_H(x)+\partial_H(y)$		D5
$\partial_H(\underline{0}) = \underline{0}$		D2DR	$\partial_H(\sigma.x) = \sigma.\partial_H(x)$		D6DR
$\partial_H(\underline{a}.x) = \underline{0}$	if $a\in H$	D3DR	$\partial_H(\sigma^*x) = \sigma^*\partial_H(x)$		D7DR
$\partial_H(\underline{a}.x) = \underline{a}.\partial_H(x)$	otherwise	D4DR			

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

$\underline{0}\between x = \underline{0}$	LM1DR
$\underline{1}\between x = \underline{0}$	LM2DR
$\underline{a}.x\between y = \underline{a}.(x\parallel y)$	LM3DR
$(x+y)\between z = x\between z + y\between z$	LM4
$\sigma.x\between\nu(y) = \underline{0}$	LM5DR
$\sigma.x\between(\nu(y)+\sigma.z) = \sigma.(x\between z)$	LM6DR
$\sigma^*x\between\sigma^*\nu(y) = \sigma^*(x\between\sigma^*\nu(y))$	LM7DR

$x\mid y = y\mid x$	SC1	$(x\parallel y)\parallel z = x\parallel(y\parallel z)$	SC4
$x\parallel 1 = x$	SC2	$(x\mid y)\mid z = x\mid(y\mid z)$	SC5
$\underline{1}\mid x+\underline{1} = \underline{1}$	SC3DR	$(x\between y)\between z = x\between(y\parallel z)$	SC6
$x\cdot 1\between 0 = x\cdot 0$	SC8DR	$(x\mid y)\between z = x\mid(y\between z)$	SC7

$\underline{0}\mid x = \underline{0}$		CM1DR
$(x+y)\mid z = x\mid z + y\mid z$		CM2
$\underline{1}\mid\underline{1} = \underline{1}$		CM3DR
$\underline{a}.x\mid\underline{1} = \underline{0}$		CM4DR
$\underline{a}.x\mid\underline{b}.y = \underline{c}.(x\parallel y)$	if $\gamma(a,b) = c$	CM5DR
$\underline{a}.x\mid\underline{b}.y = \underline{0}$	if $\gamma(a,b)$ not defined	CM6DR
$\sigma.x\mid\nu(y) = \underline{0}$		CM7DR
$\sigma.x\mid\sigma.y = \sigma.(x\mid y)$		CM8DR
$\sigma^*x\mid\sigma^*y = \sigma^*(x\mid\sigma^*y+\sigma^*x\mid y)$		CM9DR

Table 9.5. The process theory $\mathrm{TCP}^{\mathrm{drt}*}(A,\gamma)$ (with $a, b, c \in A$).

Proposition 9.7.1 (Identities in $\mathrm{TCP}^{\mathrm{drt}*}(A,\gamma)$) For any $\mathrm{TCP}^{\mathrm{drt}*}(A,\gamma)$-terms x, y, action $a\in A$, and $H\subseteq A$,

(i) $\mathrm{TCP}^{\mathrm{drt}*}(A,\gamma)\vdash \nu(0) = \underline{0}$;

https://doi.org/10.1017/CBO9781139195003.013 Published online by Cambridge University Press

(ii) $\mathrm{TCP}^{\mathrm{drt}^*}(A, \gamma) \vdash \nu(1) = \underline{1}$;
(iii) $\mathrm{TCP}^{\mathrm{drt}^*}(A, \gamma) \vdash \nu(a.x) = \underline{a}.x$;
(iv) $\mathrm{TCP}^{\mathrm{drt}^*}(A, \gamma) \vdash \nu(\sigma^* x) = \nu(x)$;
(v) $\mathrm{TCP}^{\mathrm{drt}^*}(A, \gamma) \vdash \partial_H(0) = 0$;
(vi) $\mathrm{TCP}^{\mathrm{drt}^*}(A, \gamma) \vdash \partial_H(1) = 1$;
(vii) for $a \in H$, $\mathrm{TCP}^{\mathrm{drt}^*}(A, \gamma) \vdash \partial_H(a.x) = 0$;
(viii) for $a \notin H$, $\mathrm{TCP}^{\mathrm{drt}^*}(A, \gamma) \vdash \partial_H(a.x) = a.\partial_H(x)$;
(ix) $\mathrm{TCP}^{\mathrm{drt}^*}(A, \gamma) \vdash 0 \cdot x = 0$;
(x) $\mathrm{TCP}^{\mathrm{drt}^*}(A, \gamma) \vdash 1 \cdot x = \sigma^* x$;
(xi) $\mathrm{TCP}^{\mathrm{drt}^*}(A, \gamma) \vdash a.x \cdot y = a.(x \cdot y)$;
(xii) $\mathrm{TCP}^{\mathrm{drt}^*}(A, \gamma) \vdash \sigma^* x + 0 = \sigma^* x$.

Proof See Exercise 9.7.1. □

The axioms for parallel composition and the auxiliary parallel-composition operators are such that parallel processes can only delay if both components allow this delay. Within each time slice, parallel processes interleave their actions or communicate. To stay as closely as possible to the interpretation of the axioms in the untimed setting, it is necessary for both left merge and communication merge to synchronize passage of time as well (Axioms LM6DR and CM8DR). The empty process 1 is still the identity element of parallel composition (Axiom SC2). Some axioms of the untimed theory are no longer valid in full generality, e.g., $0 \between x = 0$ (LM1) is not valid for all processes x (take e.g. $\underline{0}$ for x), but only for processes that allow an initial arbitrary delay, i.e., *delayable* processes in the sense of Axioms DD, DT, and DA of Table 9.3. Delayable processes are processes that can be written in the form $\sigma^* \nu(y)$. Axiom ATS implies that the progress of time does not affect a delayable process. The following proposition gives some identities concerning parallel-composition operators that can be derived from $\mathrm{TCP}^{\mathrm{drt}^*}(A, \gamma)$. Most of these identities are directly derived from axioms of the untimed theory $\mathrm{TCP}(A, \gamma)$. Some of them are in fact identical to axioms from this theory, whereas others are axioms reformulated for delayable processes only.

Proposition 9.7.2 (Identities in $\mathrm{TCP}^{\mathrm{drt}^*}(A, \gamma)$) For arbitrary $\mathrm{TCP}^{\mathrm{drt}^*}(A, \gamma)$-terms x, y and actions $a, b, c \in A$,

(i) $\mathrm{TCP}^{\mathrm{drt}^*}(A, \gamma) \vdash \sigma.x \parallel \sigma.y = \sigma.(x \parallel y)$;
(ii) $\mathrm{TCP}^{\mathrm{drt}^*}(A, \gamma) \vdash 0 \between \sigma^* \nu(x) = 0$;
(iii) $\mathrm{TCP}^{\mathrm{drt}^*}(A, \gamma) \vdash 1 \between \sigma^* \nu(x) = 0$;
(iv) $\mathrm{TCP}^{\mathrm{drt}^*}(A, \gamma) \vdash a.x \between \sigma^* \nu(y) = a.(x \parallel \sigma^* \nu(y))$;
(v) $\mathrm{TCP}^{\mathrm{drt}^*}(A, \gamma) \vdash 0 \mid \sigma^* \nu(x) = 0$;

https://doi.org/10.1017/CBO9781139195003.013 Published online by Cambridge University Press

(vi) $\mathrm{TCP}^{\mathrm{drt}^*}(A,\gamma) \vdash 1 \mid 1 = 1$;
(vii) $\mathrm{TCP}^{\mathrm{drt}^*}(A,\gamma) \vdash 1 \parallel 1 = 1$;
(viii) $\mathrm{TCP}^{\mathrm{drt}^*}(A,\gamma) \vdash a.x \mid 1 = 0$;
(ix) $\mathrm{TCP}^{\mathrm{drt}^*}(A,\gamma) \vdash a.x \mid b.y = c.(x \parallel y)$ if $\gamma(a,b) = c$;
(x) $\mathrm{TCP}^{\mathrm{drt}^*}(A,\gamma) \vdash a.x \mid b.y = 0$ if $\gamma(a,b)$ is not defined;
(xi) $\mathrm{TCP}^{\mathrm{drt}^*}(A,\gamma) \vdash 1 \mid \sigma^*\nu(x) + 1 = 1$.

Proof See Exercise 9.7.2. □

The first property of the next proposition shows that terms over the signature of the untimed process theory $\mathrm{TCP}(A,\gamma)$ of Section 7.7 are delayable. For untimed processes, delayability can be expressed in several different means, as illustrated by this property. The second property in the proposition shows that an untimed process also allows arbitrary passage of time at the end. In combination with the previous two propositions, Proposition 9.7.3 implies that the axioms of the untimed theory are in the timed setting still valid for terms of the untimed theory (Exercise 9.7.4).

Proposition 9.7.3 (Delayability of untimed processes) For any arbitrary closed $\mathrm{TCP}(A,\gamma)$-term p,

$$\mathrm{TCP}^{\mathrm{drt}^*}(A,\gamma) \vdash p = 1 \cdot p = \sigma^* p = \sigma^*\nu(p)$$

and

$$\mathrm{TCP}^{\mathrm{drt}^*}(A,\gamma) \vdash p = p \cdot 1.$$

Proof See Exercise 9.7.3. □

As expected, the operators that are new in $\mathrm{TCP}^{\mathrm{drt}^*}(A,\gamma)$ when compared to $\mathrm{BSP}^{\mathrm{drt}^*}(A)$ can be eliminated.

Theorem 9.7.4 (Elimination) For any closed $\mathrm{TCP}^{\mathrm{drt}^*}(A,\gamma)$-term p, there is a closed $\mathrm{BSP}^{\mathrm{drt}^*}(A)$-term q such that $\mathrm{TCP}^{\mathrm{drt}^*}(A,\gamma) \vdash p = q$.

Proof It suffices to prove that for any closed $\mathrm{TCP}^{\mathrm{drt}^*}(A,\gamma)$-term p, there is a *basic* $\mathrm{BSP}^{\mathrm{drt}^*}(A)$-term q (see Definition 9.5.1) such that $\mathrm{TCP}^{\mathrm{drt}^*}(A,\gamma) \vdash p = q$. The proof does not use rewriting, as the axioms of $\mathrm{TCP}^{\mathrm{drt}^*}(A,\gamma)$ are not in a convenient form for such a proof. Instead, the proof uses induction. (Note that also Exercise 4.5.4 requests an induction-based proof of an elimination theorem.) The following properties are needed:

(i) for any basic $\mathrm{BSP}^{\mathrm{drt}^*}(A)$-term p, there exists a basic $\mathrm{BSP}^{\mathrm{drt}^*}(A)$-term q such that $\mathrm{TCP}^{\mathrm{drt}^*}(A,\gamma) \vdash \nu(p) = q$;

https://doi.org/10.1017/CBO9781139195003.013 Published online by Cambridge University Press

(ii) for any basic $\mathrm{BSP}^{\mathrm{drt}^*}(A)$-term p and any $H \subseteq A$, there exists a basic term q such that $\mathrm{TCP}^{\mathrm{drt}^*}(A,\gamma) \vdash \partial_H(p) = q$;

(iii) for any basic terms p and p', there exists a basic term q such that $\mathrm{TCP}^{\mathrm{drt}^*}(A,\gamma) \vdash p \cdot p' = q$;

(iv) for any basic terms p and p', there exists a basic term q such that $\mathrm{TCP}^{\mathrm{drt}^*}(A,\gamma) \vdash p \between p' = q$;

(v) for any basic terms p and p', there exists a basic term q such that $\mathrm{TCP}^{\mathrm{drt}^*}(A,\gamma) \vdash p \mid p' = q$;

(vi) for any basic terms p and p', there exists a basic term q such that $\mathrm{TCP}^{\mathrm{drt}^*}(A,\gamma) \vdash p \parallel p' = q$.

The first three properties are proven by induction on the structure of basic term p. For the third, it is necessary to consider the form of p' in case p is of the form $\sigma^n 1$, and use the tenth item of Proposition 9.7.1 (Identities in $\mathrm{TCP}^{\mathrm{drt}^*}(A,\gamma)$) together with the axioms of time iteration.

The last three properties are proven simultaneously by induction on the number of symbols of p and p'. For the first of these three cases, if p' is an alternative composition, it is necessary to consider several subcases.

The proof can now be completed by applying properties (i)–(vi) to eliminate all the six operators that are new in $\mathrm{TCP}^{\mathrm{drt}^*}(A,\gamma)$ when compared to $\mathrm{BSP}^{\mathrm{drt}^*}(A)$, starting with the smallest subterm(s) containing any of the new operators and gradually working outwards. The details of the proof are left for Exercise 9.7.13. □

A term model can be constructed along the same lines as before, except that it is necessary to introduce a class of auxiliary operators. It is not known whether a term deduction system for $\mathrm{TCP}^{\mathrm{drt}^*}(A,\gamma)$ exists without such auxiliary operators. Although not strictly necessary, the auxiliary operators are introduced in the equational theory first. For each natural number n, the *shift* operator $n \gg \_$ shifts a process in time by n time slices. Applying the $n \gg \_$ operator to a process results in the behavior that remains after n units of time have passed. Table 9.6 presents theory $\mathrm{TCP}^{\mathrm{drt}^*} \gg (A,\gamma)$, $\mathrm{TCP}^{\mathrm{drt}^*}(A,\gamma)$ with shift operators.

As an example of a derivation in $\mathrm{TCP}^{\mathrm{drt}^*} \gg (A,\gamma)$, consider the following: $\mathrm{TCP}^{\mathrm{drt}^*} \gg (A,\gamma) \vdash (n+1) \gg \sigma^*(\underline{a}.\underline{1}+\sigma.\underline{0}) = (n+1) \gg \sigma^*\nu(\underline{a}.\underline{1})+(n+1) \gg \sigma^*\sigma.\underline{0} = \sigma^*\nu(\underline{a}.\underline{1})+(n+1) \gg \sigma.\sigma^*\underline{0} = \sigma^*\underline{a}.\underline{1}+n \gg \sigma^*\underline{0} = \sigma^*\underline{a}.\underline{1}+\sigma^*\underline{0} = \sigma^*(\underline{a}.\underline{1}+\underline{0}) = \sigma^*\underline{a}.\underline{1} = a.\underline{1}$.

At this point, a term model for the extended theory $\mathrm{TCP}^{\mathrm{drt}^*} \gg (A,\gamma)$ can be constructed in the standard way. The resulting model is also a model for theory $\mathrm{TCP}^{\mathrm{drt}^*}(A,\gamma)$. The term deduction system $TDS(\mathrm{TCP}^{\mathrm{drt}^*} \gg (A,\gamma))$ consists of

$\_\_\ \mathrm{TCP}^{\mathrm{drt}*} \gg (A, \gamma)$ _____
$\mathrm{TCP}^{\mathrm{drt}*}(A, \gamma)$;

unary: $(n \gg \_)_{n \in \mathbf{N}}$;

x, y;

$0 \gg x = x$	SH1
$n \gg \underline{0} = \underline{0}$	SH2
$(n+1) \gg \underline{1} = \underline{0}$	SH3
$(n+1) \gg \underline{a}.x = \underline{0}$	SH4
$n \gg (x + y) = n \gg x + n \gg y$	SH5
$(n+1) \gg \sigma.x = n \gg x$	SH6
$n \gg \sigma^* \nu(x) = \sigma^* \nu(x)$	SH7

Table 9.6. Axioms for $\mathrm{TCP}^{\mathrm{drt}*}(A, \gamma)$ shift operators ($a \in A, n \in \mathbf{N}$).

the deduction rules from $TDS(\mathrm{BSP}^{\mathrm{drt}*}(A))$ and additionally the deduction rules given in Table 9.7.

The rules for sequential composition and encapsulation are straightforward. The 'now' operator ν only allows activity in the current time slice, so with time-stamp 0. For parallel composition, in order for termination or communication to occur, both components have to allow this in the same time slice. If a component executes an action itself, the other component must be able to reach the time slice in which the action occurs; after this action, the other component must be updated to that time slice. This is achieved via a shift operator. Operational rules for this operator are straightforward.

The algebra $\mathbb{P}(\mathrm{TCP}^{\mathrm{drt}*} \gg (A, \gamma)) = (\mathcal{C}(\mathrm{TCP}^{\mathrm{drt}*} \gg (A, \gamma)), +, \cdot, \|, \between, \mid, (n \gg \_)_{n \in \mathbf{N}}, (\underline{a}.\_)_{a \in A}, \sigma.\_, (a.\_)_{a \in A}, \sigma^*, \nu, (\partial_H)_{H \subseteq A}, \underline{0}, \underline{1}, 0, 1)$ is the term algebra for theory $\mathrm{TCP}^{\mathrm{drt}*} \gg (A, \gamma)$.

Proposition 9.7.5 (Congruence) Timed bisimilarity is a congruence on the term algebra $\mathbb{P}(\mathrm{TCP}^{\mathrm{drt}*} \gg (A, \gamma))$.

Proof The term deduction system is in path format, so congruence follows immediately; see the proof of Theorem 9.4.3 for additional explanation. □

Definition 9.7.6 (Term model of $\mathrm{TCP}^{\mathrm{drt}*} \gg (A, \gamma)$) The term model of theory $\mathrm{TCP}^{\mathrm{drt}*} \gg (A, \gamma)$ is the quotient algebra $\mathbb{P}(\mathrm{TCP}^{\mathrm{drt}*} \gg (A, \gamma))_{/\leftrightarrows_t}$.

Theorem 9.7.7 (Soundness) Theory $\mathrm{TCP}^{\mathrm{drt}*} \gg (A, \gamma)$ is a sound axiomatization of $\mathbb{P}(\mathrm{TCP}^{\mathrm{drt}*} \gg (A, \gamma))_{/\leftrightarrows_t}$, i.e., $\mathbb{P}(\mathrm{TCP}^{\mathrm{drt}*} \gg (A, \gamma))_{/\leftrightarrows_t} \models \mathrm{TCP}^{\mathrm{drt}*} \gg (A, \gamma)$.

https://doi.org/10.1017/CBO9781139195003.013 Published online by Cambridge University Press

$TDS(\mathrm{TCP}^{\mathrm{drt}*} \gg (A,\gamma))$
$TDS(\mathrm{BSP}^{\mathrm{drt}*}(A))$;

unary: ν, $(\partial_H)_{H\subseteq A}$, $(n \gg \_)_{n\in\mathbf{N}}$; binary: $\_\cdot\_$, $\_\parallel\_$, $\_\between\_$, $\_\mid\_$;

x, x', y, y';

$$\frac{x\downarrow_n \quad y\downarrow_m}{x\cdot y\downarrow_{n+m}} \qquad \frac{x\ {}_n\!\xrightarrow{a} x'}{x\cdot y\ {}_n\!\xrightarrow{a} x'\cdot y} \qquad \frac{x\downarrow_n \quad y\ {}_m\!\xrightarrow{a} y'}{x\cdot y\ {}_{n+m}\!\xrightarrow{a} y'}$$

$$\frac{x\rightsquigarrow_n}{x\cdot y\rightsquigarrow_n} \qquad \frac{x\downarrow_n \quad y\rightsquigarrow_m}{x\cdot y\rightsquigarrow_{n+m}}$$

$$\frac{x\downarrow_0}{\nu(x)\downarrow_0} \qquad \frac{x\ {}_0\!\xrightarrow{a} x'}{\nu(x)\ {}_0\!\xrightarrow{a} x'}$$

$$\frac{x\downarrow_n}{\partial_H(x)\downarrow_n} \qquad \frac{x\ {}_n\!\xrightarrow{a} x' \quad a\notin H}{\partial_H(x)\ {}_n\!\xrightarrow{a} x'} \qquad \frac{x\rightsquigarrow_n}{\partial_H(x)\rightsquigarrow_n}$$

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

$$\frac{x\ {}_n\!\xrightarrow{a} x' \quad y\rightsquigarrow_n}{x\parallel y\ {}_n\!\xrightarrow{a} x'\parallel(n\gg y)} \qquad \frac{x\rightsquigarrow_n \quad y\ {}_n\!\xrightarrow{a} y'}{x\parallel y\ {}_n\!\xrightarrow{a} (n\gg x)\parallel y'} \qquad \frac{x\ {}_n\!\xrightarrow{a} x' \quad y\rightsquigarrow_n}{x\between y\xrightarrow{a} x'\parallel(n\gg y)}$$

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

$$\frac{x\rightsquigarrow_n \quad y\rightsquigarrow_n}{x\parallel y\rightsquigarrow_n} \qquad \frac{x\rightsquigarrow_n \quad y\rightsquigarrow_n}{x\between y\rightsquigarrow_n} \qquad \frac{x\rightsquigarrow_n \quad y\rightsquigarrow_n}{x\mid y\rightsquigarrow_n}$$

$$\frac{x\ {}_{n+m}\!\xrightarrow{a} x'}{n\gg x\ {}_m\!\xrightarrow{a} x'} \qquad \frac{x\downarrow_{n+m}}{n\gg x\downarrow_m} \qquad \frac{x\rightsquigarrow_{n+m}}{n\gg x\rightsquigarrow_m}$$

Table 9.7. Term deduction system for $\mathrm{TCP}^{\mathrm{drt}*} \gg (A,\gamma)$ (with $a,b,c \in A$, $n,m \in \mathbf{N}$, and $H \subseteq A$).

Proof See Exercise 9.7.14. □

Theorem 9.7.8 (Ground-completeness) Theory $\mathrm{TCP}^{\mathrm{drt}*} \gg (A,\gamma)$ is a complete axiomatization of the term model $\mathbb{P}(\mathrm{TCP}^{\mathrm{drt}*} \gg (A,\gamma))_{/\leftrightarrows_t}$, i.e., for any closed $\mathrm{TCP}^{\mathrm{drt}*} \gg (A,\gamma)$-terms p and q, $\mathbb{P}(\mathrm{TCP}^{\mathrm{drt}*} \gg (A,\gamma))_{/\leftrightarrows_t} \models p = q$ implies $\mathrm{TCP}^{\mathrm{drt}*} \gg (A,\gamma) \vdash p = q$.

https://doi.org/10.1017/CBO9781139195003.013 Published online by Cambridge University Press

Proof The proof goes along the usual lines, see e.g., Theorem 6.3.9 (Ground-completeness of TSP(A)), but using the generalized operational framework with transition systems with multiple types of transitions and/or state predicates of (Baeten & Verhoef, 1995). □

To end this section, it is interesting to observe that the process theory developed in this section is a conservative ground-extension with respect to earlier timed and untimed theories.

Theorem 9.7.9 (Conservative ground-extension) Theory $\mathrm{TCP}^{\mathrm{drt}^*} \gg (A, \gamma)$ is a conservative ground-extension of theory $\mathrm{BSP}^{\mathrm{drt}^*}(A)$.

Proof The theorem follows from results for the generalized operational framework with transition systems with multiple types of transitions and/or state predicates of (Baeten & Verhoef, 1995). □

Theorem 9.7.10 (Conservative ground-extension) Theory $\mathrm{TCP}^{\mathrm{drt}^*} \gg (A, \gamma)$ is a conservative ground-extension of theory $\mathrm{TCP}(A, \gamma)$.

Proof The proof uses a generalization of Proposition 9.6.1 (Timed vs. strong bisimilarity) to closed $\mathrm{TCP}(A, \gamma)$-terms, and then goes along the same lines as Theorem 9.6.2 (Conservative ground-extension). Alternative proofs use the meta-theory of (Baeten *et al.*, 2005) or the result of Exercise 9.7.4. □

Exercises

9.7.1 Prove Proposition 9.7.1 (Identities in $\mathrm{TCP}^{\mathrm{drt}^*}(A, \gamma)$).

9.7.2 Prove Proposition 9.7.2 (Identities in $\mathrm{TCP}^{\mathrm{drt}^*}(A, \gamma)$).

9.7.3 Prove Proposition 9.7.3 (Delayability of untimed processes).

9.7.4 Prove that, for all closed $\mathrm{TCP}(A, \gamma)$-terms, all axioms of the untimed process theory $\mathrm{TCP}(A, \gamma)$ are derivable from the axioms of $\mathrm{TCP}^{\mathrm{drt}^*}(A, \gamma)$.

9.7.5 Prove that

$$\mathrm{TCP}^{\mathrm{drt}^*}(A, \gamma) \vdash \nu(\partial_H(p)) = \partial_H(\nu(p)),$$

for all $H \subseteq A$ and all closed $\mathrm{TCP}^{\mathrm{drt}^*}(A, \gamma)$-terms p.

9.7.6 Prove that

$$\mathrm{TCP}^{\mathrm{drt}^*}(A, \gamma) \vdash \underline{1} \mid p = \nu(\partial_A(p)),$$

for all closed $\mathrm{TCP}^{\mathrm{drt}^*}(A, \gamma)$-terms p.

https://doi.org/10.1017/CBO9781139195003.013 Published online by Cambridge University Press

9.7.7 Prove that in the process theory obtained from $\mathrm{TCP}^{\mathrm{drt}*}(A,\gamma)$ by removing axioms $\underline{1} \mid \underline{1} = \underline{1}$ and $\underline{a}.x \mid \underline{1} = \underline{0}$, and adding axiom $\underline{1} \mid x = \nu(\partial_A(x))$ the removed identities are derivable.

9.7.8 Prove that

$$\mathrm{TCP}^{\mathrm{drt}*}(A,\gamma) \vdash 1 \mid p = \partial_A(p),$$

for all closed $\mathrm{TCP}^{\mathrm{drt}*}(A,\gamma)$-terms p.

9.7.9 Prove that the following identities are derivable from $\mathrm{TCP}^{\mathrm{drt}*}(A,\gamma)$ for all $\mathrm{TCP}^{\mathrm{drt}*}(A,\gamma)$-terms x, y:

(a) $1 \cdot 1 = 1$;
(b) $1 \cdot \sigma.x = \sigma.1 \cdot x$;
(c) $1 \cdot (x + y) = 1 \cdot x + 1 \cdot y$.

9.7.10 Establish whether the following identities are derivable from theory $\mathrm{TCP}^{\mathrm{drt}*}(A,\gamma)$; if so, give a derivation, if not, give a counterexample.

(a) $\sigma^*x \between \sigma^*y = \sigma^*(\sigma^*x \between \sigma^*y)$;
(b) $\sigma^*x \mid \sigma^*y = \sigma^*(\sigma^*x \mid \sigma^*y)$;
(c) $\sigma^*x \parallel \sigma^*y = \sigma^*(\sigma^*x \parallel \sigma^*y)$.

9.7.11 Establish whether the following identities are derivable from theory $\mathrm{TCP}^{\mathrm{drt}*}(A,\gamma)$ for closed terms p and q; if so, give a proof, if not, give a counterexample.

(a) $\sigma^*p \between \nu(q) = \nu(p) \between \nu(q)$;
(b) $\sigma^*p \mid \nu(q) = \nu(p) \mid \nu(q)$;
(c) $\sigma^*p \between \sigma^*q = \sigma^*(p \between \sigma^*q)$;
(d) $\sigma^*\nu(p) \mid \sigma^*\nu(q) = \sigma^*(\nu(p) \mid \nu(q))$.

9.7.12 Prove that the identity

$$\sigma^*(x \mid y) = \sigma^*x \mid \sigma^*y$$

is not derivable from the axioms of $\mathrm{TCP}^{\mathrm{drt}*}(A,\gamma)$.

9.7.13 Prove Theorem 9.7.4 (Elimination).

9.7.14 Prove Theorem 9.7.7 (Soundness).

9.8 Fischer's protocol

Fischer's protocol (Lamport, 1987) is a well known mutual-exclusion protocol for timed processes. This section describes the protocol using the theory $\mathrm{TCP}^{\mathrm{drt}*}(A,\gamma)$ extended with recursion. The protocol is linearized, i.e., written in the form of a linear recursive specification, and the timed transition system corresponding to the protocol is given. Recursion has not been introduced formally in the previous sections of this chapter. Recursion can be added similarly

https://doi.org/10.1017/CBO9781139195003.013 Published online by Cambridge University Press

as in the untimed theory, be it that the time-prefix operator $\sigma_{\_}$ should also be considered as a guard for recursion variables. On the other hand, time iteration cannot be considered a guard.

Mutual exclusion is relevant in a context where processes have so-called critical sections. The goal of a mutual-exclusion protocol is to guarantee that at any time at most one of the processes, called protocol entities from now to distinguish them from other processes, is in a critical section, in combination with the requirement that at least one of the protocol entities is able to proceed at each moment in time.

The protocol entities that use Fischer's protocol to guarantee mutual exclusion make use of a shared variable to exchange information. A protocol entity that wants to enter its critical section, checks if the value of the shared variable is 0, which represents that no entity is trying to enter the critical section. Then, it assigns its unique identifier to the shared variable. In case the variable still has this value after some time, it decides to enter the critical section. Otherwise, it will try again later. The delay that is introduced before an entity enters a critical section causes the protocol entity to wait sufficiently long so that other protocol entities that *concurrently* assigned their unique identifiers have had time enough to do so. This is necessary to guarantee the mutual-exclusion property.

In the variant of Fischer's protocol that is described in this section, only two protocol entities are considered for which the unique identifiers 1 and 2 are used. The protocol entities can perform two actions with respect to the shared variable x. The first action is inspection of the value of the variable; the second is the assignment of a value to the variable. For the description in this section, it is assumed that both the assignment of a value to the shared variable and the testing of the shared variable for a specific value take no time, i.e., occur instantaneously.

In $\mathrm{TCP}^{\mathrm{drt}*}(A, \gamma)$, the shared variable x is modeled by a set of recursion variables X_v, where v denotes the value of x. The actions of the two entities with respect to the shared variable are modeled by communication. The shared-variable process is at all times willing to send its value to the environment with the action $!(x = v)$. Of course this has no effect on the value of the variable. On the other hand, the shared-variable process receives assignments to the variable by means of the action $?(x := w)$, where w is some value. Of course the value of the variable is adapted accordingly. Finally, the variable may terminate at any time. This behavior is described by the following recursive equation:

$$X_v = 1 + !(x = v).X_v + \sum_{w \in \{0,1,2\}} ?(x := w).X_w.$$

https://doi.org/10.1017/CBO9781139195003.013 Published online by Cambridge University Press

An equivalent, but from the viewpoint of manipulation by means of the axioms more convenient specification of the variable is the following, where the any-time-slice termination and the any-time-slice action prefixes are removed and all initial time passage is combined into one summand:

$$X_v = \underline{1} + \underline{!(x = v)}.X_v + \sum_{w \in \{0,1,2\}} \underline{?(x := w)}.X_w + \sigma.X_v.$$

The two protocol entities that play a role in the version of Fischer's protocol that is described in this section have similar behavior. The only difference is that they have different identities, 1 and 2 respectively. The delay period between setting the value of the shared variable and testing the shared variable for this same value, is taken to be one time unit, which is sufficient given the assumption that writing a value to the shared variable is instantaneous. Furthermore, once a protocol entity has entered its critical section, it can stay there for any amount of time. The recursive specifications for the protocol entities are as follows. Recursion variable A describes protocol entity 1 and recursion variable B describes protocol entity 2.

$$\begin{aligned} A = {} & 1 + ?(x = 0).\underline{!(x := 1)}.\sigma. \\ & (\ \underline{?(x = 0)}.A \\ & + \underline{?(x = 1)}.\underline{enterCS_1}.leaveCS_1.\underline{!(x := 0)}.A \\ & + \underline{?(x = 2)}.A \\ &) \ , \end{aligned}$$

$$\begin{aligned} B = {} & 1 + ?(x = 0).\underline{!(x := 2)}.\sigma. \\ & (\ \underline{?(x = 0)}.B \\ & + \underline{?(x = 1)}.B \\ & + \underline{?(x = 2)}.\underline{enterCS_2}.leaveCS_2.\underline{!(x := 0)}.B \\ &) \ . \end{aligned}$$

The whole system is given by the recursion variable FP with the following recursive specification:

$$FP = \partial_H(A \parallel X_0 \parallel B),$$

where for all $\alpha \in \{x = i, x := i \mid i \in \{0, 1, 2\}\}$

$$\gamma(!\alpha, ?\alpha) = \gamma(?\alpha, !\alpha) = ?\alpha$$

and γ is undefined otherwise, and where

$$H = \{!\alpha, ?\alpha \mid \alpha \in \{x = i, x := i \mid i \in \{0, 1, 2\}\}\}.$$

Using the axioms, a linear version of Fischer's protocol is obtained easily. First, linear versions of the protocol entities themselves are presented. The recursion variables A_i describe protocol entity 1 and the recursion variables B_i

https://doi.org/10.1017/CBO9781139195003.013 Published online by Cambridge University Press

describe protocol entity 2. It can be proven that $(\mathrm{TCP}_{\mathrm{rec}}^{\mathrm{drt}^*} + \mathrm{RSP})(A, \gamma) \vdash A = A_0$ and $(\mathrm{TCP}_{\mathrm{rec}}^{\mathrm{drt}^*} + \mathrm{RSP})(A, \gamma) \vdash B = B_0$.

$$
\begin{aligned}
A_0 &= \underline{1} + \underline{?(x = 0)}.A_1 + \sigma.A_0\\
A_1 &= \underline{!(x := 1)}.A_2\\
A_2 &= \sigma.A_3\\
A_3 &= \underline{?(x = 0)}.A_0 + \underline{?(x = 1)}.A_4 + \underline{?(x = 2)}.A_0\\
A_4 &= \underline{\mathit{enterCS}_1}.A_5\\
A_5 &= \underline{\mathit{leaveCS}_1}.A_6 + \sigma.A_5\\
A_6 &= \underline{!(x := 0)}.A_0
\end{aligned}
$$

$$
\begin{aligned}
B_0 &= \underline{1} + \underline{?(x = 0)}.B_1 + \sigma.B_0\\
B_1 &= \underline{!(x := 2)}.B_2\\
B_2 &= \sigma.B_3\\
B_3 &= \underline{?(x = 0)}.B_0 + \underline{?(x = 1)}.B_0 + \underline{?(x = 2)}.B_4\\
B_4 &= \underline{\mathit{enterCS}_2}.B_5\\
B_5 &= \underline{\mathit{leaveCS}_2}.B_6 + \sigma.B_5\\
B_6 &= \underline{!(x := 0)}.B_0
\end{aligned}
$$

Linearizing process FP, which is derivably equal to $\partial_H(A_0 \parallel X_0 \parallel B_0)$, results in the following 32 recursive equations. The recursion variable S_{ijk} corresponds to the process term $\partial_H(A_i \parallel X_j \parallel B_k)$, which means that $FP \equiv S_{000}$.

$$
\begin{aligned}
S_{000} &= \partial_H(A_0 \parallel X_0 \parallel B_0)\\
&= \underline{1} + \underline{?(x = 0)}.\partial_H(A_1 \parallel X_0 \parallel B_0)\\
&\quad + \underline{?(x = 0)}.\partial_H(A_0 \parallel X_0 \parallel B_1) + \sigma.\partial_H(A_0 \parallel X_0 \parallel B_0)\\
&= \underline{1} + \underline{?(x = 0)}.S_{100} + \underline{?(x = 0)}.S_{001} + \sigma.S_{000}
\end{aligned}
$$

$$
\begin{aligned}
S_{100} &= \partial_H(A_1 \parallel X_0 \parallel B_0)\\
&= \underline{?(x := 1)}.\partial_H(A_2 \parallel X_1 \parallel B_0) + \underline{?(x = 0)}.\partial_H(A_1 \parallel X_0 \parallel B_1)\\
&= \underline{?(x := 1)}.S_{210} + \underline{?(x = 0)}.S_{101}
\end{aligned}
$$

$$
\begin{aligned}
S_{001} &= \partial_H(A_0 \parallel X_0 \parallel B_1)\\
&= \underline{?(x = 0)}.\partial_H(A_1 \parallel X_0 \parallel B_1) + \underline{?(x := 2)}.\partial_H(A_0 \parallel X_2 \parallel B_2)\\
&= \underline{?(x = 0)}.S_{101} + \underline{?(x := 2)}.S_{022}
\end{aligned}
$$

$$
\begin{aligned}
S_{210} &= \partial_H(A_2 \parallel X_1 \parallel B_0)\\
&= \sigma.\partial_H(A_3 \parallel X_1 \parallel B_0)\\
&= \sigma.S_{310}
\end{aligned}
$$

$$
\begin{aligned}
S_{101} &= \partial_H(A_1 \parallel X_0 \parallel B_1)\\
&= \underline{?(x := 1)}.\partial_H(A_2 \parallel X_1 \parallel B_1) + \underline{?(x := 2)}.\partial_H(A_1 \parallel X_2 \parallel B_2)\\
&= \underline{?(x := 1)}.S_{211} + \underline{?(x := 2)}.S_{122}
\end{aligned}
$$

https://doi.org/10.1017/CBO9781139195003.013 Published online by Cambridge University Press

$$\begin{aligned} S_{022} &= \partial_H(A_0 \parallel X_2 \parallel B_2) \\ &= \sigma.\partial_H(A_0 \parallel X_2 \parallel B_3) \\ &= \sigma.S_{023} \end{aligned}$$

$$\begin{aligned} S_{310} &= \partial_H(A_3 \parallel X_1 \parallel B_0) \\ &= \underline{?(x=1)}.\partial_H(A_4 \parallel X_1 \parallel B_0) \\ &= \underline{?(x=1)}.S_{410} \end{aligned}$$

$$\begin{aligned} S_{211} &= \partial_H(A_2 \parallel X_1 \parallel B_1) \\ &= \underline{?(x:=2)}.\partial_H(A_2 \parallel X_2 \parallel B_2) \\ &= \underline{?(x:=2)}.S_{222} \end{aligned}$$

$$\begin{aligned} S_{122} &= \partial_H(A_1 \parallel X_2 \parallel B_2) \\ &= \underline{?(x:=1)}.\partial_H(A_2 \parallel X_1 \parallel B_2) \\ &= \underline{?(x:=1)}.S_{212} \end{aligned}$$

$$\begin{aligned} S_{023} &= \partial_H(A_0 \parallel X_2 \parallel B_3) \\ &= \underline{?(x=2)}.\partial_H(A_0 \parallel X_2 \parallel B_4) \\ &= \underline{?(x=2)}.S_{024} \end{aligned}$$

$$\begin{aligned} S_{410} &= \partial_H(A_4 \parallel X_1 \parallel B_0) \\ &= \underline{enterCS_1}.\partial_H(A_5 \parallel X_1 \parallel B_0) \\ &= \underline{enterCS_1}.S_{510} \end{aligned}$$

$$\begin{aligned} S_{222} &= \partial_H(A_2 \parallel X_2 \parallel B_2) \\ &= \sigma.\partial_H(A_3 \parallel X_2 \parallel B_3) \\ &= \sigma.S_{323} \end{aligned}$$

$$\begin{aligned} S_{212} &= \partial_H(A_2 \parallel X_1 \parallel B_2) \\ &= \sigma.\partial_H(A_3 \parallel X_1 \parallel B_3) \\ &= \sigma.S_{313} \end{aligned}$$

$$\begin{aligned} S_{024} &= \partial_H(A_0 \parallel X_2 \parallel B_4) \\ &= \underline{enterCS_2}.\partial_H(A_0 \parallel X_2 \parallel B_5) \\ &= \underline{enterCS_2}.S_{025} \end{aligned}$$

$$\begin{aligned} S_{510} &= \partial_H(A_5 \parallel X_1 \parallel B_0) \\ &= \underline{leaveCS_1}.\partial_H(A_6 \parallel X_1 \parallel B_0) + \sigma.\partial_H(A_5 \parallel X_1 \parallel B_0) \\ &= \underline{leaveCS_1}.S_{610} + \sigma.S_{510} \end{aligned}$$

$$\begin{aligned} S_{323} &= \partial_H(A_3 \parallel X_2 \parallel B_3) \\ &= \underline{?(x=2)}.S_{023} + \underline{?(x=2)}.S_{324} \end{aligned}$$

$$\begin{aligned} S_{313} &= \partial_H(A_3 \parallel X_1 \parallel B_3) \\ &= \underline{?(x=1)}.S_{413} + \underline{?(x=1)}.S_{310} \end{aligned}$$

https://doi.org/10.1017/CBO9781139195003.013 Published online by Cambridge University Press

$$
\begin{aligned}
S_{025} &= \partial_H(A_0 \parallel X_2 \parallel B_5) \\
&= \underline{leaveCS_2}.S_{026} + \sigma.S_{025}
\end{aligned}
$$

$$
\begin{aligned}
S_{610} &= \partial_H(A_6 \parallel X_1 \parallel B_0) \\
&= \underline{?(x := 0)}.S_{000}
\end{aligned}
$$

$$
\begin{aligned}
S_{324} &= \partial_H(A_3 \parallel X_2 \parallel B_4) \\
&= \underline{?(x = 2)}.S_{024} + \underline{enterCS_2}.S_{325}
\end{aligned}
$$

$$
\begin{aligned}
S_{413} &= \partial_H(A_4 \parallel X_1 \parallel B_3) \\
&= \underline{enterCS_1}.S_{513} + \underline{?(x = 1)}.S_{410}
\end{aligned}
$$

$$
\begin{aligned}
S_{026} &= \partial_H(A_0 \parallel X_2 \parallel B_6) \\
&= \underline{?(x := 0)}.S_{000}
\end{aligned}
$$

$$
\begin{aligned}
S_{325} &= \partial_H(A_3 \parallel X_2 \parallel B_5) \\
&= \underline{?(x = 2)}.S_{025} + \underline{leaveCS_2}.S_{326}
\end{aligned}
$$

$$
\begin{aligned}
S_{513} &= \partial_H(A_5 \parallel X_1 \parallel B_3) \\
&= \underline{leaveCS_1}.S_{613} + \underline{?(x = 1)}.S_{510}
\end{aligned}
$$

$$
\begin{aligned}
S_{326} &= \partial_H(A_3 \parallel X_2 \parallel B_6) \\
&= \underline{?(x = 2)}.S_{026} + \underline{?(x := 0)}.S_{300}
\end{aligned}
$$

$$
\begin{aligned}
S_{613} &= \partial_H(A_6 \parallel X_1 \parallel B_3) \\
&= \underline{?(x := 0)}.S_{003} + \underline{?(x = 1)}.S_{610}
\end{aligned}
$$

$$
\begin{aligned}
S_{300} &= \partial_H(A_3 \parallel X_0 \parallel B_0) \\
&= \underline{?(x = 0)}.S_{000} + \underline{?(x = 0)}.S_{301}
\end{aligned}
$$

$$
\begin{aligned}
S_{003} &= \partial_H(A_0 \parallel X_0 \parallel B_3) \\
&= \underline{?(x = 0)}.S_{103} + \underline{?(x = 0)}.S_{000}
\end{aligned}
$$

$$
\begin{aligned}
S_{301} &= \partial_H(A_3 \parallel X_0 \parallel B_1) \\
&= \underline{?(x = 0)}.S_{001} + \underline{?(x := 2)}.S_{322}
\end{aligned}
$$

$$
\begin{aligned}
S_{103} &= \partial_H(A_1 \parallel X_0 \parallel B_3) \\
&= \underline{?(x := 1)}.S_{213} + \underline{?(x = 0)}.S_{100}
\end{aligned}
$$

$$
\begin{aligned}
S_{322} &= \partial_H(A_3 \parallel X_2 \parallel B_2) \\
&= \underline{?(x = 0)}.S_{022}
\end{aligned}
$$

$$
\begin{aligned}
S_{213} &= \partial_H(A_2 \parallel X_1 \parallel B_3) \\
&= \underline{?(x = 1)}.S_{210}
\end{aligned}
$$

Figure 9.3 gives the timed transition system corresponding to the process term *FP*. The ? symbol is omitted from the labels. Also the enter and leave labels are simplified. Whenever a transition has a natural number n as a label,

https://doi.org/10.1017/CBO9781139195003.013 Published online by Cambridge University Press

this means that there is a transition for every natural number. By careful inspection of the timed transition system, it can easily be established that indeed there is no possibility to enter a critical section in case the other critical section has already been entered and not yet left.

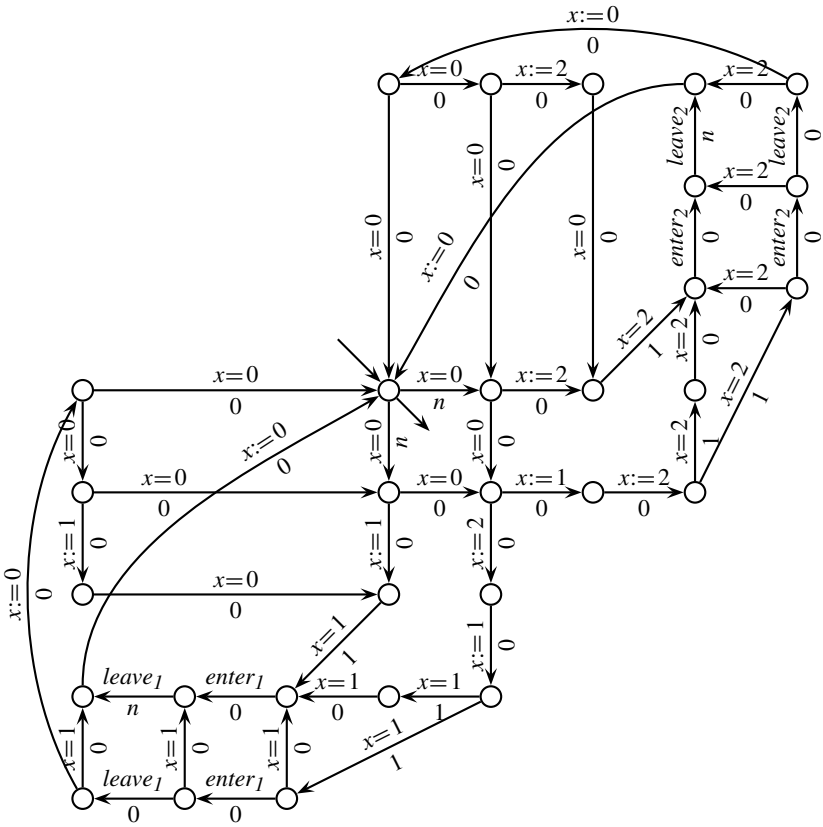


Fig. 9.3. Timed transition system for Fischer's protocol.

9.9 Bibliographical remarks

This chapter finds its origin in (Baeten & Bergstra, 1996). Further development took place in (Baeten, 2003; Baeten & Middelburg, 2002; Baeten & Reniers, 2004; Baeten *et al.*, 2005). Notation follows these references, except that this book uses single underlining instead of double underlining for relative-time theories.

The operational semantics as it is given in this chapter is based on (Baeten

https://doi.org/10.1017/CBO9781139195003.013 Published online by Cambridge University Press

& Reniers, 2004). The treatment of parallel composition is from (Baeten & Reniers, 2007).

The notion of embedding mentioned in Section 9.3 for the relation between the untimed process theory BSP(A)and the timed process theory $\mathrm{BSP}^{\mathrm{drt}}(A)$ is formally defined in (Baeten & Middelburg, 2001).

Essentially, the presentation of Fischer's protocol in Section 9.8 is taken from (Vereijken, 1997). The only difference is that here the protocol entities can stay in their critical sections an arbitrary amount of time, whereas in (Vereijken, 1997) a critical section is left in the same time slice as it was entered.

Closest in syntax elements to the present timed process theory is the work on timed extensions of CSP, see (Reed & Roscoe, 1988; Schneider, 2000). For timed extensions of CCS, see e.g., (Moller & Tofts, 1990; Yi, 1991; Hennessy & Regan, 1995). A comparison is made in (Corradini *et al.*, 1999). Other related timed process theories are ATP, see (Nicollin & Sifakis, 1994), and timed LOTOS, see for example (Quemada *et al.*, 1993).

10

Data and states

10.1 Introduction

In the previous chapters, data types have been handled in an informal way. An alternative composition parameterized by a finite data type D, written as $\sum_{d \in D} t$ with t some process term possibly containing d, was introduced as an abbreviation of a finite expression, and the d occurring in term t was not treated as a (bound) variable. This chapter takes a closer look at data expressions. Such expressions are considered in a more formal way, and the interplay between data and processes is studied. All issues involved can be illustrated by considering just two concrete data types, namely the (finite) data type of the Booleans and the (infinite) data type of the natural numbers. The data type of the natural numbers was also used in the previous chapter to denote time behavior. Considering an uncountable data type as the reals causes additional problems that are avoided in the present text. The use of an uncountable data type in a parameterized alternative composition would, just like the use of an uncountable time domain, provide a means to specify uncountable processes, which is not possible with any of the theories developed in this book.

Notation 10.1.1 (Booleans, propositional logic) Recall from Example 2.3.2 the algebra of the Booleans $\mathbb{B} = (\mathbf{B}, \wedge, \neg, \mathit{true})$. In addition to the constant *true* and the operators $\wedge$ and $\neg$, the binary operators $\vee$ (or) and $\supset$ (implication), and the constant *false* are also used in the remainder. The not so common symbol $\supset$ is used for implication in order to avoid the use of too many arrows in notations.

Let $\mathbf{P} = \{P_1, \ldots, P_n\}$ for some natural number n be a set of so-called propositional variables. Later on, specific instances of these variables are given. Starting from the propositional variables, the Boolean constants and the operators on the Booleans introduced above, it is possible to build terms along the lines of Definition 2.2.3 (Terms). These terms are referred to as propositional

https://doi.org/10.1017/CBO9781139195003.014 Published online by Cambridge University Press

terms or propositional logic formulas, and this set of formulas is denoted **FB**. Examples of propositional formulas are $P_1 \wedge \mathit{true}$ and $\neg(P_1 \vee P_2)$. Given specific (Boolean) values for P_1 and P_2, these formulas evaluate either to *true* or to *false*.

Besides a more detailed consideration of data types, this chapter also takes a closer look at the notion of a state. In process algebra, a common assumption is that states are not observable, and the notion of bisimilarity considers unnamed states. Nevertheless, in some cases it is desirable to have certain aspects of a state to be observable. The chapter describes mechanisms in order to realize this. Observable aspects of a state can typically be expressed by means of a propositional logic formula, which links the two main concepts investigated in this chapter, data types and states.

10.2 Guarded commands

The most straightforward connection between data and processes is the use of conditionals. A conditional can be introduced as a constant, or as a unary or binary operator on process terms. In each case, the operator is parameterized by a propositional term. The presentation in this section considers conditionals as unary operators, called the *guarded-command* operators. Given a propositional formula ϕ, the guarded command corresponding to ϕ applied to term x is written as $\phi :\rightarrow x$, with the intuitive meaning '*if* ϕ *then* x'. The extension of basic process theory BSP(A) with guarded commands, called (BSP + GC)(A), is given in Table 10.1. Axioms GC1–6 are mostly self-explanatory.

$\_\_$ (BSP + GC)(A) $\_\_\_\_\_$
BSP(A);
unary: $(\phi :\rightarrow \_)_{\phi \in \mathbf{FB}}$;
x, y;

$\mathit{true} :\rightarrow x = x$	GC1
$\mathit{false} :\rightarrow x = 0$	GC2
$\phi :\rightarrow 0 = 0$	GC3
$\phi :\rightarrow (x + y) = (\phi :\rightarrow x) + (\phi :\rightarrow y)$	GC4
$(\phi \vee \psi) :\rightarrow x = (\phi :\rightarrow x) + (\psi :\rightarrow x)$	GC5
$\phi :\rightarrow (\psi :\rightarrow x) = (\phi \wedge \psi) :\rightarrow x$	GC6

Table 10.1. Process theory (BSP + GC)(A) (with $\phi, \psi \in \mathbf{FB}$).

In the remainder, assume, as before, that the unary guarded-command operators bind stronger than binary operators; assume that they bind weaker than other unary operators.

https://doi.org/10.1017/CBO9781139195003.014 Published online by Cambridge University Press

A guarded-command operator blocks further progress if the guard evaluates to false, and does nothing (skips) if the guard evaluates to true. An expression of the form *if* ϕ *then* x *else* y, for propositional formula ϕ and process terms x and y, can be represented by the term $\phi :\to x + \neg\phi :\to y$. Guarded commands are similar to encapsulation operators (progress is blocked based on the identity of actions) and projection operators (progress is blocked based on the number of actions that have been executed). However, there are two important differences with these types of operators. First of all, the guarded command applies only to the first action; it disappears as soon as one action is executed. In fact, the absence of an axiom for action-prefix operators in Table 10.1 implies that an action in the context of a guarded command can never be executed without resolving the guard. Second, a guarded command can also block termination ($\mathit{false} :\to 1 = 0$), whereas encapsulation or projection cannot prevent termination.

An interesting observation is that Axiom GC5, in combination with Axioms GC1 and GC2, causes Axioms A3 and A6 of the basic theory BSP(A) to be derivable (see Exercise 10.2.1).

In line with the above discussion, an expression as $P :\to a.0$, with P a propositional variable and a an action, cannot be simplified unless the truth value of P is known. As a consequence, there is no elimination theorem as long as there are (unknown) propositional variables.

Related to this last observation is the fact that axioms concerning guarded commands, such as Axioms GC4 and GC5, are in derivations typically used from right to left. In this way, the scope of a conditional is enlarged as much as possible, so that the terms within the scope and/or the signals are amenable to simplification. As a very simple example, consider the following derivation, with x a process term, $a \in A$, and $\phi \in \mathbf{FB}$:

$$\begin{aligned}&(\mathrm{BSP} + \mathrm{GC})(A) \vdash\\ &\quad a.(\phi :\to x + \neg\phi :\to x) = a.((\phi \vee \neg\phi) :\to x) =\\ &\qquad a.(\mathit{true} :\to x) = a.x.\end{aligned}$$

In order to give an operational semantics, it is important to note that it is needed to know the values of the propositional variables in order to decide on possible transitions. In realistic processes, furthermore, values of propositional variables can change during the execution of a process. For instance, it can be the case that the value of a propositional variable P is *true* if and only if exactly two actions have been executed. To capture the values of propositional variables in the operational framework of transition-system spaces, it is necessary to associate valuations of the propositional variables, that is, functions v from $\mathbf{P}$ to $\mathbf{B} = \{\mathit{true}, \mathit{false}\}$ to the states of such a space. The set of these valuations

https://doi.org/10.1017/CBO9781139195003.014 Published online by Cambridge University Press

is denoted BV. Note that every valuation can easily be extended to a function from propositional formulas in **FB** to **B**. Upon executing an action a in a state with valuation v, a state with a possibly different valuation v' results. The resulting valuation v' is called the *effect* of the execution of action a in a state with valuation v. The effect of action execution can be captured by the following function:

$$\mathit{effect} : A \times BV \to BV.$$

The *effect* function is a parameter of the operational semantics of a process theory with guarded commands. As in the previous chapter, the notion of a transition-system space needs to be redefined. In the current context, it is needed to integrate valuations of propositional variables as explained above. A transition-system space over a set of states is equipped with the following predicates and relations:

- Predicates $\langle \_, v \rangle \downarrow$ for each $v \in BV$;
- Relations $\langle \_, v \rangle \xrightarrow{a} \langle \_, \mathit{effect}(a, v) \rangle$ for each $v \in BV, a \in A$.

The above notation emphasizes that termination predicates and action relations are defined for the combinations of states and valuations. An alternative notation that is in use attaches valuations to the predicate and transition arrows, leading for example to $s \overset{v,a,v'}{\to} s'$ instead of $\langle s, v \rangle \xrightarrow{a} \langle s', v' \rangle$ (with s and s' states and $v' = \mathit{effect}(a, v)$). This last notation suggests that any valuation-action-effect triple can be seen as a (structured) action itself.

Table 10.2 gives the term deduction system underlying the standard term model for theory $(\mathrm{BSP} + \mathrm{GC})(A)$. Several interesting observations can be made with respect to this term deduction system. First, as already mentioned, the effect function *effect* is a parameter of the term deduction system, which means that the transition systems associated to terms depend on this function. Second, as usual, only $(\mathrm{BSP}+\mathrm{GC})(A)$-terms are considered that do not contain process variables. However, process terms may have propositional variables in guards, because the deduction rules for the guarded-command operators evaluate the guards. $(\mathrm{BSP} + \mathrm{GC})(A)$-terms thus do not need to be closed with respect to propositional variables, but only with respect to process variables. This implies that, for a given *effect* function, the transition system associated to a $(\mathrm{BSP}+\mathrm{GC})(A)$-term that is closed with respect to process variables captures the behavior for *all possible* valuations of the propositional variables *at any point* during the execution of the process.

Example 10.2.1 (Transition systems of $(\mathrm{BSP} + \mathrm{GC})(A)$-terms) Let P be a propositional variable, and assume for simplicity that it is the only propositional variable. This means that there are two valuations, v_t with $v_t(P) = \mathit{true}$

https://doi.org/10.1017/CBO9781139195003.014 Published online by Cambridge University Press

$TDS((\mathrm{BSP} + \mathrm{GC})(A), \mathit{effect})$
constant: 0, 1; unary: $(a._-)_{a\in A}$, $(\phi :\rightarrow \_)_{\phi\in\mathbf{FB}}$; binary: $\_ + \_$;
x, x', y, y';

$$\langle a.x, v\rangle \xrightarrow{a} \langle x, \mathit{effect}(a, v)\rangle \qquad \langle 1, v\rangle\downarrow$$

$$\frac{\langle x, v\rangle \xrightarrow{a} \langle x', v'\rangle}{\langle x + y, v\rangle \xrightarrow{a} \langle x', v'\rangle} \qquad \frac{\langle y, v\rangle \xrightarrow{a} \langle y', v'\rangle}{\langle x + y, v\rangle \xrightarrow{a} \langle y', v'\rangle}$$

$$\frac{\langle x, v\rangle\downarrow}{\langle x + y, v\rangle\downarrow} \qquad \frac{\langle y, v\rangle\downarrow}{\langle x + y, v\rangle\downarrow}$$

$$\frac{\langle x, v\rangle \xrightarrow{a} \langle x', v'\rangle \quad v(\phi) = \mathit{true}}{\langle \phi :\rightarrow x, v\rangle \xrightarrow{a} \langle x', v'\rangle} \qquad \frac{\langle x, v\rangle \downarrow \quad v(\phi) = \mathit{true}}{\langle \phi :\rightarrow x, v\rangle \downarrow}$$

Table 10.2. Term deduction system for (BSP+GC)(A) (with $\phi \in \mathbf{FB}$, $a \in A$, $v, v' \in BV$).

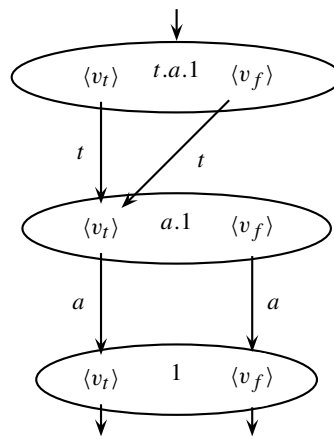


Fig. 10.1. The transition system corresponding to $t.a.1$.

and v_f with $v_f(P) = \mathit{false}$. Thus, $BV = \{v_t, v_f\}$. Let t be an action in A such that for both valuations $v \in BV$, $\mathit{effect}(t, v) = v_t$. Assume that action a does not change a valuation, i.e., $\mathit{effect}(a, v) = v$ for both $v \in BV$.

Consider term $t.a.1$. Figure 10.1 shows the transition system corresponding to this term. The figure visualizes that transitions and termination depend on the valuation of propositional variables.

Figure 10.1 shows that the transition system of a term captures the behavior for all possible valuations for the propositional variables. By assuming an

https://doi.org/10.1017/CBO9781139195003.014 Published online by Cambridge University Press

initial valuation, a concrete transition system for that situation can be obtained. For example, when assuming v_t as the initial valuation, term $t.a.1$ results in a transition system of three states with only two transitions. When v_f is the initial valuation, the transition system also has three states and two transitions. The first transition of this transition system changes the valuation from v_f to v_t.

One could wonder whether it is relevant to capture transition $\langle a.1, v_f \rangle \xrightarrow{a} \langle 1, v_f \rangle$ in the transition system. It is clear that this transition can never occur, irrespective of the valuation of the propositional variable P in the initial state. Nevertheless, it is important to capture also this behavior, as becomes clear later on when considering equivalence of processes. As already mentioned, a transition system in the current framework captures the behavior for *all possible* valuations of the propositional variables *in all states* that may occur during the execution of the process.

Another aspect that is worth illustrating is the impact of the *effect* function on state transitions. Assume, for example, that also action t has no effect on the valuation of propositional variable P. This would result in a change of transition $\langle t.a.1, v_f \rangle \xrightarrow{a} \langle a.1, v_t \rangle$ into $\langle t.a.1, v_f \rangle \xrightarrow{a} \langle a.1, v_f \rangle$.

Finally, consider the term $t.(P :\rightarrow a.1)$, assuming the original definition for the *effect* function. Despite the fact that this term has a propositional variable, the term deduction system of Table 10.2 associates a transition system with it. The transition system, shown in Figure 10.2, is very similar to the transition system of Figure 10.1. It has different terms associated with the top two states, and the rightmost transition between the bottom two states does not exist, i.e., $\langle P :\rightarrow a.1, v_f \rangle \overset{a}{\not\rightarrow}$. It is interesting to observe that, irrespective of the initial valuation of the propositional variable, the observable transitions of the two transition systems in Figures 10.1 and 10.2 are identical.

The introduction of propositional variables and valuations of those variables has led to an adapted notion of transition-system spaces. It is also necessary to reconsider the notion of bisimilarity. The following example illustrates that it is important to require that two processes can only be bisimilar if they behave the same in all states for all possible valuations of the propositional variables in those states.

Example 10.2.2 (Guarded commands and equivalence of processes) Consider again terms $t.a.1$ and $t.(P :\rightarrow a.1)$ of Example 10.2.1, and their transition systems given in Figures 10.1 and 10.2. As already explained, when considering any given specific initial valuation of the propositional variable P, these transition systems behave the same. Nevertheless, $t.a.1 = t.(P :\rightarrow a.1)$

https://doi.org/10.1017/CBO9781139195003.014 Published online by Cambridge University Press

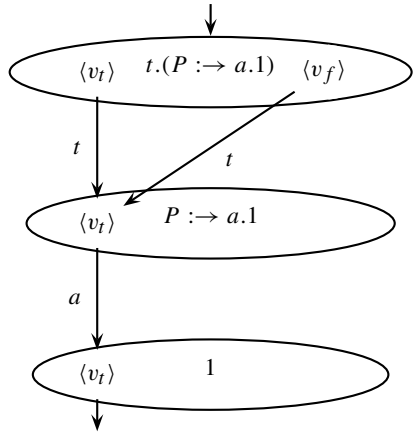


Fig. 10.2. The transition system corresponding to $t.(P :\rightarrow a.1)$.

cannot be derived from theory (BSP + GC)(A), for the simple reason that the theory does not have any axioms to reason in general about processes with propositional variables. As a consequence, the guarded command in the right-hand term cannot be resolved, and it cannot be eliminated.

This apparent mismatch is intentional. The mentioned pair of terms and the processes they define should not be identified. The reason for this becomes clear when considering the extension with parallel composition. Consider again term $t.(P :\rightarrow a.1)$. After execution of t, propositional variable P is *true*, but by activity in a parallel component it might be that P has turned *false* again when action a is attempted, effectively blocking the execution of a. The a action cannot be blocked in the process specified by term $t.a.1$. As an example, assume the mentioned processes are running in parallel with the process $f.1$, where f is an action in A such that for all valuations $v \in BV$, $\mathit{effect}(f, v) = v_f$ (with v_f the valuation defined in Example 10.2.1 that sets P to *false*). The resulting processes $t.a.1 \parallel f.1$ and $t.(P :\rightarrow a.1) \parallel f.1$ are not equivalent (see Exercise 10.2.3).

By defining an appropriate notion of bisimilarity on the transition systems generated by the operational semantics, it can be ensured that pairs of terms as the one discussed in this example are not bisimilar. The essential point is that all possible valuations of propositional variables should be considered in all states of a process.

Definition 10.2.3 (Bisimilarity) Assume a transition-system space with states S over the set of labels A. A binary relation R on the set of states S is a *bisimulation* relation if and only if the following transfer conditions hold:

(i) for all states $s, t, s' \in S$, whenever $(s, t) \in R$ and $\langle s, v\rangle \xrightarrow{a} \langle s', v'\rangle$ for some $a \in A$ and $v \in BV$ (implying $v' = \mathit{effect}(a, v)$), then there is a state t' such that $\langle t, v\rangle \xrightarrow{a} \langle t', v'\rangle$ and $(s', t') \in R$;

(ii) vice versa, for all states $s, t, t' \in S$, whenever $(s, t) \in R$ and $\langle t, v\rangle \xrightarrow{a} \langle t', v'\rangle$ for some $a \in A$ and $v \in BV$, then there is a state s' such that $\langle s, v\rangle \xrightarrow{a} \langle s', v'\rangle$ and $(s', t') \in R$;

(iii) whenever $(s, t) \in R$ and $\langle s, v\rangle\downarrow$ for some $v \in BV$, then $\langle t, v\rangle\downarrow$;

(iv) whenever $(s, t) \in R$ and $\langle t, v\rangle\downarrow$ for some $v \in BV$, then $\langle s, v\rangle\downarrow$.

Two transition systems $s, t \in S$ are *bisimilar*, denoted by the standard notation as $s \leftrightarrows t$, if and only if there is a bisimulation relation R on S with $(s, t) \in R$.

At this point, it is possible to define a term model for theory (BSP + GC)(A) along the usual lines. The term algebra consists of the set of (BSP + GC)(A)-terms that are closed with respect to process variables, but not necessarily with respect to propositional variables. The latter is consistent with the fact that the terms define processes whose behavior may depend on the value of propositional variables, of which the value is determined by the initial valuation and the effect of action execution on this valuation. Bisimilarity as defined above is a congruence relation on the resulting algebra of transition systems induced by the term deduction system in Table 10.2. Theory (BSP + GC)(A) is a sound and ground-complete axiomatization of the term model obtained as the quotient algebra of this algebra of transition systems. Note that this model is parameterized with the *effect* function. The soundness and ground-completeness results hold for *any effect* function and all terms that are closed with respect to process variables. The theory thus allows to prove the equivalence of processes when they behave the same for all possible *effect* functions, all possible initial valuations of propositional variables, and all possible valuations of those variables in any intermediate state. It is not possible to reason about the equivalence of processes for specific *effect* functions and initial valuations. Section 10.5 introduces a family of operators that makes it possible to reason about processes for specific *effect* functions and initial valuations.

Theory (BSP + GC)(A) is a conservative ground-extension of the basic theory BSP(A). One way to prove this is along the lines of the proof of the conservativity result given in Section 9.6, Theorem 9.6.2. This proof uses the fact that two closed BSP(A)-terms are bisimilar in the current setting if and only if they are strongly bisimilar as defined in the standard framework of Chapter 3. This last result follows from the observations that, in the underlying transition-system space as introduced in this section, successful termination for a closed BSP(A)-term is independent of the particular valuation of the propositional

variables and that all transitions with the same action label originating from a given state in the transition-system space end up in the same state.

Extension of theory (BSP + GC)(A) to larger theories, including the extension with recursion, does not present problems. Table 10.3 gives theory (TCP + GC)(A, γ), which extends earlier theories (conservatively with respect to closed terms) with axioms concerning the interplay between guarded commands on the one hand, and sequential composition, encapsulation, or parallel-composition operators on the other hand. In combination with Axiom SC1, Axiom GC9, read from right to left, states that communication can only occur if conditions guarding any of the communicating processes are satisfied.

Consideration should be given to the description of communication. If $\gamma(a, b) = c$, then *effect*(c, v) should somehow denote the joint effect of the execution of a and b given some valuation v. If the effects of a and b are contradictory, then they should not be able to communicate, and so c cannot be executed. In line with the earlier observation that theory (BSP + GC)(A) can only be used to reason about the equivalence of processes that are equivalent for all valuations of propositional variables in all states of the process, it is necessary to require that a communication between actions a and b can only occur if the effects of these two actions are consistent for all valuations. Formally, for any pair of actions a and b, $\gamma(a, b)$ is defined only if *effect*$(b,$ *effect*$(a, v)) =$ *effect*$(a,$ *effect*$(b, v)) =$ *effect*$(\gamma(a, b), v)$ for all valuations $v \in BV$. Table 10.4 shows the deduction rules of the operational semantics of (TCP+GC)(A, γ) that are related to communication. For further details, see Exercise 10.2.5.

(TCP + GC)(A, γ)
(BSP + GC)(A), TCP(A, γ);
-
x, y;

$\phi :\rightarrow (x \cdot y) = (\phi :\rightarrow x) \cdot y$	GC7
$\phi :\rightarrow (x \between y) = (\phi :\rightarrow x) \between y$	GC8
$\phi :\rightarrow (x \mid y) = (\phi :\rightarrow x) \mid y$	GC9
$\phi :\rightarrow \partial_H(x) = \partial_H(\phi :\rightarrow x)$	GC10

Table 10.3. The process theory (TCP + GC)(A, γ) (with $\phi \in \mathbf{FB}$, $H \subseteq A$).

Example 10.2.4 (Guarded commands) Consider a description of the behavior of a spring. There are actions *pull*, *release* and *break*, and propositional variables *extended* and *malfunction*. Action *pull* takes the variable *extended* from *false* to *true*, and action *release* takes this value back from *true* to *false*.

https://doi.org/10.1017/CBO9781139195003.014 Published online by Cambridge University Press

$TDS((\mathrm{TCP}+\mathrm{GC})(A,\gamma),\mathit{effect})$
$TDS((\mathrm{BSP}+\mathrm{GC})(A),\mathit{effect})$;
unary: $(\partial_H)_{H\subseteq A}$; binary: $\_\cdot\_, \_\parallel\_, \_\between\_, \_\mid\_$;
x, x', y, y';

$$\frac{\langle x,v\rangle \xrightarrow{a} \langle x',v'\rangle \quad \langle y,v\rangle \xrightarrow{b} \langle y',v''\rangle \quad \gamma(a,b)=c}{\langle x\parallel y,v\rangle \xrightarrow{c} \langle x'\parallel y',\mathit{effect}(c,v)\rangle}$$

$$\frac{\langle x,v\rangle \xrightarrow{a} \langle x',v'\rangle \quad \langle y,v\rangle \xrightarrow{b} \langle y',v''\rangle \quad \gamma(a,b)=c}{\langle x\mid y,v\rangle \xrightarrow{c} \langle x'\parallel y',\mathit{effect}(c,v)\rangle}$$

$\ldots$

Table 10.4. Some operational rules from the term deduction system for $(\mathrm{TCP}+\mathrm{GC})(A,\gamma)$ (with $a,b,c\in A$, $v,v',v''\in BV$).

Action *break* can only occur when the spring is extended, and causes a malfunction, changing the value of *malfunction* from *false* to *true*. The spring can only be pulled again, if it has not yet malfunctioned. The spring process can terminate successfully if the spring is not extended or broken. The following recursive specification defines the process *Spring* with the use of a recursion variable X. There is no communication in this example.

$$\begin{aligned} X &= 1 + \mathit{pull}.\mathit{release}.(\neg \mathit{malfunction} :\to X)\\ \mathit{Spring} &= X \parallel (1 + \mathit{extended} :\to \mathit{break}.1). \end{aligned}$$

When assuming the initial valuation that sets propositional variables *extended* and *malfunction* to *false*, process *Spring* has the expected operational behavior (see Exercise 10.2.4).

Exercises

10.2.1 Derive Axioms A3 and A6 from the other axioms of $(\mathrm{BSP}+\mathrm{GC})(A)$.

10.2.2 Draw the transition system for term $t.(P :\to a.1 + \neg P :\to b.0)$. Assume that P, t, and a are as defined in Example 10.2.1 (Transition systems of $(\mathrm{BSP}+\mathrm{GC})(A)$-terms), and assume that b has no effect on the value of propositional variable P. Establish, using Definition 10.2.3 (Bisimilarity), that the transition system is not bisimilar to the transition system of term $t.a.1$, illustrated in Figure 10.1.

10.2.3 Consider the terms $t.a.1 \parallel f.1$ and $t.(P :\to a.1) \parallel f.1$ discussed in Example 10.2.2 (Guarded commands and equivalence of processes), and

https://doi.org/10.1017/CBO9781139195003.014 Published online by Cambridge University Press

assume that there is no communication. Draw the transition systems for these two terms, and establish that they are not bisimilar.

10.2.4 Draw the transition system of the spring in Example 10.2.4 (Guarded commands) for the concrete initial valuation that sets both variables *extended* and *malfunction* to *false*. Give for each state the variables that are true in that state.

10.2.5 Complete the term deduction system given in Table 10.4 for theory $(\mathrm{TCP}+\mathrm{GC})(A,\gamma)$. Prove a soundness and ground-completeness result. Prove that all operators new in $(\mathrm{TCP}+\mathrm{GC})(A,\gamma)$ compared to $(\mathrm{BSP}+\mathrm{GC})(A)$ can be eliminated and show that $(\mathrm{TCP}+\mathrm{GC})(A,\gamma)$ is a conservative ground-extension of $(\mathrm{BSP}+\mathrm{GC})(A)$.

10.3 The inaccessible process

In the course of the explorations performed in the remainder of this chapter, inconsistency of a state can be encountered. This section therefore introduces a new constant $\bot$ that denotes an inaccessible state, a state that cannot be entered by the execution of an action. Intuitively, this process denotes a state where *false* holds. As *false* can never hold, this is a state that a process can never get into. Sometimes, this process is called (rather contradictorily) the non-existent process. Here, the name *inaccessible process* is used. The theory $\mathrm{BSP}_\bot(A)$ extends the theory $\mathrm{BSP}(A)$ with the extra constant $\bot$ and adds two axioms. Table 10.5 gives theory $\mathrm{BSP}_\bot(A)$, $\mathrm{BSP}(A)$ with the inaccessible process. Axiom IP1 explains that in an inaccessible state, it does not matter which extra options are available, as the state will remain inaccessible; Axiom IP2 states that an inaccessible state cannot be entered by executing an action, as the action will be blocked in this case. $\mathrm{BSP}_\bot(A)$ is a conservative ground-extension of $\mathrm{BSP}(A)$; the inaccessible process cannot be eliminated.

$\mathrm{BSP}_\bot(A)$	
$\mathrm{BSP}(A)$;	
constant: $\bot$;	
x;	
$x+\bot=\bot$	IP1
$a.\bot=0$	IP2

Table 10.5. The process theory $\mathrm{BSP}_\bot(A)$ (with $a\in A$).

The term model for the basic process theory $\mathrm{BSP}_\bot(A)$ is omitted. Instead,

https://doi.org/10.1017/CBO9781139195003.014 Published online by Cambridge University Press

an operational semantics is given for an equational theory combining the inaccessible process and guarded commands.

Theory $(\mathrm{BSP}_\bot + \mathrm{GC})(A)$, the theory of basic sequential processes with the inaccessible process and guarded commands, simply combines the signature and axioms of theories $(\mathrm{BSP} + \mathrm{GC})(A)$ of Table 10.1 and $\mathrm{BSP}_\bot(A)$ given above. To give an operational semantics for $(\mathrm{BSP}_\bot + \mathrm{GC})(A)$, it is not only necessary to consider valuations of propositional variables, as in the previous section, but it is also necessary to take into account the consistency of states. To do so, a transition-system space as introduced in the previous section is equipped with the following additional set of predicates:

- A predicate $\langle \_, v\rangle \searrow$ for each $v \in BV$, denoting the consistency of the operand state for the given valuation.

In the transition-system space that underlies an operational semantics for a process theory with the inaccessible process $\bot$, predicates $\langle \_, v\rangle \searrow$ are needed to distinguish $\bot$ from other processes. In the current context, consistency of states does not depend on the valuation of propositional variables. All these predicates hold for all processes that do not reduce to $\bot$. Later in this chapter, consistency does depend on the valuations.

Table 10.6 gives the term deduction system underlying the standard term model for theory $(\mathrm{BSP}_\bot + \mathrm{GC})(A)$. Compared to the term deduction system given for $(\mathrm{BSP}{+}\mathrm{GC})(A)$ in Table 10.2, Table 10.6 contains extra rules defining the consistency predicates; furthermore, the deduction rules for the termination predicates and the transition relations contain extra consistency requirements in the premises, conforming to the intuition behind the inaccessible process. For example, the rule defining the transition relation for action-prefix operators requires that the process resulting after performing the action should be consistent, in line with Axiom IP2.

Because of the introduction of the consistency predicates, it is necessary to reconsider the notion of bisimilarity.

Definition 10.3.1 (Bisimilarity) Assume a transition-system space with states S over the set of labels A. A binary relation R on the set of states S is a bisimulation relation if and only if it satisfies the transfer conditions of Definition 10.2.3 and the following additional conditions:

(v) whenever $(s, t) \in R$ and $\langle s, v\rangle \searrow$ for some $v \in BV$, then $\langle t, v\rangle \searrow$;
(vi) whenever $(s, t) \in R$ and $\langle t, v\rangle \searrow$ for some $v \in BV$, then $\langle s, v\rangle \searrow$.

As before, two transition systems $s, t \in S$ are *bisimilar*, again denoted $s \leftrightarrows t$, if and only if there is a bisimulation relation R on S with $(s, t) \in R$.

https://doi.org/10.1017/CBO9781139195003.014 Published online by Cambridge University Press

$\_\_ TDS((\mathrm{BSP}_\perp + \mathrm{GC})(A), \mathit{effect}) \_\_\_\_\_$
constant: $0, 1, \perp$; unary: $(a.\_)_{a\in A}, (\phi :\to \_)_{\phi\in\mathbf{FB}}$; binary: $\_ + \_$;
x, x', y, y';

$$\langle 0, v\rangle \searrow \qquad \langle 1, v\rangle \searrow \qquad \langle 1, v\rangle\downarrow$$

$$\frac{\langle x, v'\rangle \searrow \quad v' = \mathit{effect}(a, v)}{\langle a.x, v\rangle \xrightarrow{a} \langle x, v'\rangle} \qquad \langle a.x, v\rangle \searrow$$

$$\frac{\langle x, v\rangle \xrightarrow{a} \langle x', v'\rangle \quad \langle y, v\rangle \searrow}{\langle x + y, v\rangle \xrightarrow{a} \langle x', v'\rangle} \qquad \frac{\langle y, v\rangle \xrightarrow{a} \langle y', v'\rangle \quad \langle x, v\rangle \searrow}{\langle x + y, v\rangle \xrightarrow{a} \langle y', v'\rangle}$$

$$\frac{\langle x, v\rangle\downarrow \quad \langle y, v\rangle \searrow}{\langle x + y, v\rangle\downarrow} \qquad \frac{\langle y, v\rangle\downarrow \quad \langle x, v\rangle \searrow}{\langle x + y, v\rangle\downarrow} \qquad \frac{\langle x, v\rangle \searrow \quad \langle y, v\rangle \searrow}{\langle x + y, v\rangle \searrow}$$

$$\frac{\langle x, v\rangle \searrow \quad v(\phi) = \mathit{true}}{\langle \phi :\to x, v\rangle \searrow} \qquad \frac{\langle x, v\rangle \downarrow \quad v(\phi) = \mathit{true}}{\langle \phi :\to x, v\rangle \downarrow}$$

$$\frac{\langle x, v\rangle \xrightarrow{a} \langle x', v'\rangle \quad v(\phi) = \mathit{true}}{\langle \phi :\to x, v\rangle \xrightarrow{a} \langle x', v'\rangle} \qquad \frac{v(\phi) = \mathit{false}}{\langle \phi :\to x, v\rangle \searrow}$$

Table 10.6. Term deduction system for $(\mathrm{BSP}_\perp + \mathrm{GC})(A)$ (with $\phi \in \mathbf{FB}$, $a \in A$, $v, v' \in BV$).

At this point, a term model of $(\mathrm{BSP}_\perp + \mathrm{GC})(A)$ can be constructed along the usual lines, for which $(\mathrm{BSP}_\perp + \mathrm{GC})(A)$ is a ground-complete axiomatization. As in the previous section, the *effect* function is a parameter of this model. Theory $(\mathrm{BSP}_\perp + \mathrm{GC})(A)$ does not allow any elimination results, but it is a conservative ground-extension of theories $\mathrm{BSP}_\perp(A)$ and $(\mathrm{BSP} + \mathrm{GC})(A)$.

Theory $(\mathrm{BSP}_\perp + \mathrm{GC})(A)$ can be extended to larger theories as usual. Table 10.7 gives theory $(\mathrm{TCP}_\perp + \mathrm{GC})(A, \gamma)$. As in the previous section, the communication and *effect* functions should be defined in a consistent way. Further details are left to the reader.

Exercises

10.3.1 Give an operational semantics for theory $\mathrm{BSP}_\perp(A)$, based on a transition-system space without valuation predicates but with a (single) consistency predicate.

10.3.2 Recall Proposition 5.5.14 (Guardedness). Consider theory $\mathrm{BSP}_\perp(A)$ with recursion. Prove that the unguarded recursive specification $\{X = \perp + X\}$ has only one solution, for any action set A.

https://doi.org/10.1017/CBO9781139195003.014 Published online by Cambridge University Press

$(\mathrm{TCP}_\perp + \mathrm{GC})(A,\gamma)$
$(\mathrm{TCP} + \mathrm{GC})(A,\gamma), \mathrm{BSP}_\perp(A);$
-
$x;$

$\perp \cdot x = \perp$	IP3
$\perp \between x = \perp$	IP4
$x \between \perp = \perp$	IP5
$\perp \mid x = \perp$	IP6
$\partial_H(\perp) = \perp$	IP7

Table 10.7. Theory $(\mathrm{TCP}_\perp + \mathrm{GC})(A,\gamma)$ (with $H \subseteq A$).

10.3.3 The inaccessible process $\perp$ resembles to some extent the chaos process τ^*0 discussed in Section 8.7, that led to theory $(\mathrm{BSP}^*_\tau + \mathrm{CH})(A)$ of Table 8.11 with a non-standard model developed in Exercise 8.7.6. Compare the inaccessible process and the chaos process, by investigating typical laws satisfied by these two processes.

10.4 Propositional signals

Building upon the concepts of conditionals and the inaccessible process of the previous sections, this section describes a mechanism that allows to observe aspects of the current state of a process in the equational theory. The central assumption is that the visible part of the state of a process is a proposition, an expression in propositional logic as defined in Notation 10.1.1. Such a proposition representing the visible aspects of a process state is called a *signal*. Conditionals are used to observe signals.

The introduction of the *root-signal emission operator* $\wedge\!\blacktriangle$ is done in Table 10.8. A term of the form $\phi \wedge\!\blacktriangle x$ represents the process x, that shows the signal ϕ in its initial state.

Axiom RSE1 states that any process emits a *true* signal. A process emitting a signal denotes that this signal holds in the initial state of the process. Falsity never holds, so a state emitting *false* cannot occur, is inaccessible. This explains RSE2, and shows the usefulness of the inaccessible process. (Technically, the inclusion of the inaccessible process in the theory is not strictly necessary because its role can be taken over by a term such as $\mathit{false} \wedge\!\blacktriangle 0$.) Axiom RSE3 states that the inaccessible process does not emit any signals. Note that Axiom RSE3 is derivable from the other axioms of $(\mathrm{BSP}_\perp + \mathrm{RSE})(A)$ (see Exercise 10.4.1).

Axiom RSE4 in Table 10.8 shows that the signals of a summand in an

https://doi.org/10.1017/CBO9781139195003.014 Published online by Cambridge University Press

$(\mathrm{BSP}_\perp + \mathrm{RSE})(A)$

$(\mathrm{BSP}_\perp + \mathrm{GC})(A)$;

unary: $(\phi \mathbin{\blacktriangle} \_)_{\phi \in \mathbf{FB}}$;

x, y;

$\mathit{true} \mathbin{\blacktriangle} x = x$	RSE1
$\mathit{false} \mathbin{\blacktriangle} x = \perp$	RSE2
$\phi \mathbin{\blacktriangle} \perp = \perp$	RSE3
$(\phi \mathbin{\blacktriangle} x) + y = \phi \mathbin{\blacktriangle} (x + y)$	RSE4
$\phi \mathbin{\blacktriangle} (\psi \mathbin{\blacktriangle} x) = (\phi \wedge \psi) \mathbin{\blacktriangle} x$	RSE5
$\phi :\rightarrow (\psi \mathbin{\blacktriangle} x) = (\phi \supset \psi) \mathbin{\blacktriangle} (\phi :\rightarrow x)$	RSE6
$\phi \mathbin{\blacktriangle} (\phi :\rightarrow x) = \phi \mathbin{\blacktriangle} x$	RSE7

Table 10.8. The process theory $(\mathrm{BSP}_\perp + \mathrm{RSE})(A)$ (with $\phi, \psi \in \mathbf{FB}$).

alternative composition carry over to the whole process. It can be given in a more symmetric form as follows:

$$(\phi \mathbin{\blacktriangle} x) + (\psi \mathbin{\blacktriangle} y) = (\phi \wedge \psi) \mathbin{\blacktriangle} (x + y) \,.$$

This identity depends on the fact that the roots of two processes in an alternative composition are identified. Therefore, signals must be combined. Exercise 10.4.2 shows that this generalized version of Axiom RSE4 is derivable from the theory. It is straightforward to show that the general identity implies Axiom RSE4 as given in Table 10.8. Also notice that Axiom IP1, given in Table 10.5, is derivable from RSE4.

Axiom RSE5 expresses the fact that there is no sequential order in the presentation of signals. The combination of the signals is taking both of them.

As an example, consider the following derivation:

$$\begin{aligned}&(\mathrm{BSP}_\perp + \mathrm{RSE})(A) \vdash \\ &\quad a.((\phi \mathbin{\blacktriangle} x) + (\neg\phi \mathbin{\blacktriangle} x)) = a.((\phi \wedge \neg\phi) \mathbin{\blacktriangle} x) = \\ &\qquad a.(\mathit{false} \mathbin{\blacktriangle} x) = a.\perp = 0.\end{aligned}$$

Axiom RSE6 expresses how to take a signal outside of a conditional: signal ψ is only emitted if condition ϕ is true. The last axiom, RSE7, is the *signal inspection rule*. It says that a conditional can be removed if the state emits a signal that validates the conditional. In other words, if a signal ϕ is emitted, then ϕ holds in the current state and a conditional with guard ϕ can be removed. Note that RSE7 can be generalized as follows:

$$\begin{aligned}&(\mathrm{BSP}_\perp + \mathrm{RSE})(A) \vdash \\ &\quad \phi \mathbin{\blacktriangle} ((\phi \wedge \psi) :\rightarrow x) = \phi \mathbin{\blacktriangle} (\phi :\rightarrow (\psi :\rightarrow x)) = \phi \mathbin{\blacktriangle} (\psi :\rightarrow x) \,.\end{aligned}$$

An interesting identity that follows from the theory is the following:

https://doi.org/10.1017/CBO9781139195003.014 Published online by Cambridge University Press

Proposition 10.4.1 (Signals) $(\mathrm{BSP}_\perp + \mathrm{RSE})(A) \vdash \phi \leftsquigarrow x = (\phi \leftsquigarrow 0) + x$.

Proof Exercise 10.4.2. □

The equation in Proposition 10.4.1 is very useful for writing process specifications because it allows to a large extent to work with algebraic expressions that are not cluttered with signal emissions. It provides the basis for a notion of basic terms, as introduced before in Chapters 2 and 9. Only the inaction constant emits a signal in basic terms. As conditions, only propositional formulas different from *false* are allowed. This guarantees that all actions and terminations occurring in a term can actually be executed at some point. Signal *false* is allowed as a signal emitted by the 0 constant. By doing so, it is possible to represent $\perp$ as $\mathit{false} \leftsquigarrow 0$.

Definition 10.4.2 (Basic $(\mathrm{BSP}_\perp + \mathrm{RSE})(A)$-terms) The set of basic $(\mathrm{BSP}_\perp + \mathrm{RSE})(A)$-terms is defined inductively:

(i) if $\phi \in \mathbf{FB}$, then $\phi \leftsquigarrow 0$ is a basic term;
(ii) if ϕ is not equivalent to *false*, then $\phi :\to 1$ is a basic term;
(iii) if ϕ is not equivalent to *false*, if $a \in A$ and if t is a basic $(\mathrm{BSP}_\perp + \mathrm{RSE})(A)$-term, then $\phi :\to a.t$ is a basic term;
(iv) if s, t are basic terms, then $s + t$ is a basic term.

Each basic term can be written in the form

$$\chi \leftsquigarrow 0 + \sum_{i=1}^{n} \phi_i :\to a_i.t_i + (\psi :\to 1),$$

where $n \geq 0$, where $\chi \in \mathbf{FB}$ is an arbitrary propositional formula, conditions ϕ_i and ψ are formulas all different from *false*, where all $a_i \in A$ and all t_i are basic terms, and where the termination term may or may not occur (as denoted via the parentheses).

As before, an elimination result can be proven. The proof uses Proposition 10.4.1 (Signals) and the lemma given below. Note that it is not possible to eliminate signals entirely. Theory $(\mathrm{BSP}_\perp + \mathrm{RSE})(A)$ does in general not have any means to remove signals that do not resolve to *true* or *false*.

Proposition 10.4.3 (Reduction to basic terms) For any closed $(\mathrm{BSP}_\perp + \mathrm{RSE})(A)$-term p, there exists a basic $(\mathrm{BSP}_\perp + \mathrm{RSE})(A)$-term q such that $(\mathrm{BSP}_\perp + \mathrm{RSE})(A) \vdash p = q$.

Proof Exercise 10.4.3. □

https://doi.org/10.1017/CBO9781139195003.014 Published online by Cambridge University Press

Lemma 10.4.4 $(\mathrm{BSP}_\perp + \mathrm{RSE})(A) \vdash \phi :\to \perp = \neg\phi \mathbin{\triangle} 0$.

Proof

$$(\mathrm{BSP}_\perp + \mathrm{RSE})(A) \vdash \phi :\to \perp = \phi :\to (\mathit{false} \mathbin{\triangle} 0) = (\phi \supset \mathit{false}) \mathbin{\triangle} (\phi :\to 0) = \neg\phi \mathbin{\triangle} 0\,.$$

□

$(\mathrm{BSP}_\perp + \mathrm{RSE})(A)$ is a conservative ground-extension of $(\mathrm{BSP}_\perp + \mathrm{GC})(A)$.

The next step is to look at an operational semantics. Table 10.9 presents the term deduction system for $(\mathrm{BSP}_\perp + \mathrm{RSE})(A)$, which builds upon the term deduction system of $(\mathrm{BSP}_\perp + \mathrm{GC})(A)$ that is given in Table 10.6. It has an *effect* function as a parameter, and besides transition and termination predicates, it contains a consistency predicate, as explained in the previous section. The rules are straightforward: in order to execute an action, terminate or be consistent in a certain state, the signal emitted in that state must evaluate to true. The development of a term model for which $(\mathrm{BSP}_\perp + \mathrm{RSE})(A)$ is a ground-complete axiomatization goes along the usual lines.

$$\frac{}{TDS((\mathrm{BSP}_\perp + \mathrm{RSE})(A), \mathit{effect})}$$

$$TDS((\mathrm{BSP}_\perp + \mathrm{GC})(A), \mathit{effect});$$

unary: $(\phi \mathbin{\triangle} \_)_{\phi \in \mathbf{FB}}$;

x, x';

$$\frac{\langle x, v\rangle \xrightarrow{a} \langle x', v'\rangle \quad v(\phi) = \mathit{true}}{\langle \phi \mathbin{\triangle} x, v\rangle \xrightarrow{a} \langle x', v'\rangle}$$

$$\frac{\langle x, v\rangle\downarrow \quad v(\phi) = \mathit{true}}{\langle \phi \mathbin{\triangle} x, v\rangle\downarrow} \qquad \frac{\langle x, v\rangle \searrow \quad v(\phi) = \mathit{true}}{\langle \phi \mathbin{\triangle} x, v\rangle \searrow}$$

Table 10.9. Term deduction system for $(\mathrm{BSP}_\perp + \mathrm{RSE})(A)$ (with $\phi \in \mathbf{FB}$, $a \in A$, $v, v' \in BV$).

Extensions follow the usual pattern. The next example shows a stack specification, using $(\mathrm{BSP}_\perp + \mathrm{RSE})(A)$ extended with recursion.

Example 10.4.5 (Stack) The specification of the stack from Section 5.6 can be modified with signals *empty* and *ontop*(d) for all $d \in D$ that show the top of the stack.

$$\begin{aligned} \mathit{Stack}1 &= S_\epsilon, \\ S_\epsilon &= \mathit{empty} \mathbin{\triangle} 1 + \sum_{d \in D} \mathit{push}(d).S_d, \text{ and, for all } d \in D, \sigma \in D^*, \\ S_{d\sigma} &= \mathit{ontop}(d) \mathbin{\triangle} \mathit{pop}(d).S_\sigma + \sum_{e \in D} \mathit{push}(e).S_{ed\sigma}\,. \end{aligned}$$

https://doi.org/10.1017/CBO9781139195003.014 Published online by Cambridge University Press

The extension of $(\mathrm{BSP}_\bot + \mathrm{RSE})(A)$ with sequential composition also does not present difficulties. Axioms are presented in Table 10.10, and operational rules in Table 10.11. The operational rules reflect that, if x may terminate immediately, also y must be considered in order to establish consistency of $x \cdot y$. The format of the operational deduction rules used in this book up to this point is left in the term deduction system for $(\mathrm{TSP}_\bot + \mathrm{RSE})(A)$: the last rule shows a so-called *negative premise*; the negation of a predicate is used. Nevertheless, by an extension of the theory in Chapter 3, it can be established that these rules determine a unique transition system for each closed term. The details of creating a model for which equational theory $(\mathrm{TSP}_\bot + \mathrm{RSE})(A)$ is ground-complete, are not treated here. The interested reader is referred to (Baeten & Bergstra, 1997), where such a model is developed in a slightly different operational context. For the development of meta-theory in the style of Chapter 3 that allows to establish a standard term model for which the theory is ground-complete, see (Verhoef, 1994).

$\_\_ (\mathrm{TSP}_\bot + \mathrm{RSE})(A) \_\_$
$(\mathrm{BSP}_\bot + \mathrm{RSE})(A), \mathrm{TSP}(A);$

-

$x, y;$

$\bot \cdot x = \bot$	IP3
$\phi :\rightarrow (x \cdot y) = (\phi :\rightarrow x) \cdot y$	GC7
$\phi \barwedge (x \cdot y) = (\phi \barwedge x) \cdot y$	RSE8

Table 10.10. The process theory $(\mathrm{TSP}_\bot + \mathrm{RSE})(A)$ (with $\phi \in \mathbf{FB}$).

$\_\_ TDS((\mathrm{TSP}_\bot + \mathrm{RSE})(A), \mathit{effect}) \_\_$
$TDS((\mathrm{BSP}_\bot + \mathrm{RSE})(A), \mathit{effect});$

binary: $\cdot$;

x, x', y, y' :

$$\frac{\langle x, v\rangle \xrightarrow{a} \langle x', v'\rangle \quad \langle x' \cdot y, v'\rangle \searrow}{\langle x \cdot y, v\rangle \xrightarrow{a} \langle x' \cdot y, v'\rangle} \qquad \frac{\langle x, v\rangle \downarrow \quad \langle y, v\rangle \xrightarrow{a} \langle y', v'\rangle}{\langle x \cdot y, v\rangle \xrightarrow{a} \langle y', v'\rangle}$$

$$\frac{\langle x, v\rangle \downarrow \quad \langle y, v\rangle \downarrow}{\langle x \cdot y, v\rangle \downarrow} \qquad \frac{\langle x, v\rangle \downarrow \quad \langle y, v\rangle \searrow}{\langle x \cdot y, v\rangle \searrow} \qquad \frac{\langle x, v\rangle \searrow \quad \langle x, v\rangle \not\downarrow}{\langle x \cdot y, v\rangle \searrow}$$

Table 10.11. Term deduction system for $(\mathrm{TSP}_\bot + \mathrm{RSE})(A)$ (with $a \in A$, $v, v' \in BV$).

In line with before, occurrences of the sequential-composition operator can

https://doi.org/10.1017/CBO9781139195003.014 Published online by Cambridge University Press

be eliminated from closed $(\mathrm{TSP}_\perp + \mathrm{RSE})(A)$-terms, yielding (basic) $(\mathrm{BSP}_\perp + \mathrm{RSE})(A)$-terms. The ground-extension is conservative.

With the addition of parallel composition, the mechanism of signal observation can be discussed. A process can synchronize with a process running in parallel by using conditionals that test signals emitted by the other process. This provides a new synchronization mechanism, besides the already known communication mechanism. Additional axioms are presented in Table 10.12 and explained below. As earlier in this chapter, the communication function γ needs to be defined in a way that is consistent with the *effect* function that is a parameter in the operational semantics.

$\_\_(\mathrm{BCP}_\perp + \mathrm{RSE})(A, \gamma)$
$(\mathrm{BSP}_\perp + \mathrm{RSE})(A), \mathrm{BCP}(A, \gamma)$;
unary: rs;
x, y;

$(\phi :\to x) \between y = \phi :\to (x \between y)$	GC8
$(\phi :\to x) \mid y = rs(y) + \phi :\to (x \mid y)$	GC9S
$\partial_H(\phi :\to x) = \phi :\to \partial_H(x)$	GC10
$(\phi \mathbin{/\!\blacktriangle} x) \between y = \phi \mathbin{/\!\blacktriangle} (x \between y)$	RSE9
$(\phi \mathbin{/\!\blacktriangle} x) \mid y = \phi \mathbin{/\!\blacktriangle} (x \mid y)$	RSE10
$\partial_H(\phi \mathbin{/\!\blacktriangle} x) = \phi \mathbin{/\!\blacktriangle} \partial_H(x)$	RSE11
$rs(\perp) = \perp$	RS1
$rs(1) = 0$	RS2
$rs(0) = 0$	RS3
$rs(a.x) = 0$	RS4
$rs(x + y) = rs(x) + rs(y)$	RS5
$rs(\phi :\to x) = \phi :\to rs(x)$	RS6
$rs(\phi \mathbin{/\!\blacktriangle} x) = \phi \mathbin{/\!\blacktriangle} rs(x)$	RS7

Table 10.12. The process theory $(\mathrm{BCP}_\perp + \mathrm{RSE})(A)$ (with $a \in A$, $H \subseteq A$, $\phi \in \mathbf{FB}$).

Besides the three usual parallel-composition operators and encapsulation operators needed to enforce communication, theory $(\mathrm{BCP}_\perp + \mathrm{RSE})(A)$ has one additional auxiliary operator, the root-signal operator rs, that essentially reduces a term to the signal it emits. It allows to reduce any closed term to a term of the form $\phi \mathbin{/\!\blacktriangle} 0$ for some propositional formula ϕ.

The additional RSE axioms in $(\mathrm{BCP}_\perp + \mathrm{RSE})(A)$ are self-explanatory. Note that Axiom RSE10 states that a signal emitted by one process of a pair of communicating processes is also emitted by the pair of communicating processes as a whole.

https://doi.org/10.1017/CBO9781139195003.014 Published online by Cambridge University Press

Two of the three guarded-command axioms appear in Table 10.3 in a context without signals as well (although here these axioms are presented with their left- and right-hand sides exchanged when compared to Table 10.3). Axiom GC9S replaces Axiom GC9 from Table 10.3 in a context with signals. Axiom GC9 states that a communication can only occur if conditions guarding any of the communicating processes are satisfied. Axiom GC9S states in addition that the signals emitted by any of the communicating processes remain visible at all times, even when a condition of one of the communicating processes is false (which effectively prevents communication).

Example 10.4.6 (Signal observation) A simple example of signal observation is the following. A process executing an action a that changes the value of a propositional variable P from true to false, and emitting this fact, can be specified as follows:

$$P \owedge a \,.(\neg P \owedge 1).$$

A process in parallel can observe these signals, and make progress dependent on them. Consider the following process, to be executed in parallel with the above process.

$$P :\!\rightarrow b.(\neg P :\!\rightarrow 1).$$

If in the parallel composition a is executed first, then deadlock will ensue. If b is executed first, then termination can be reached. Assume a and b cannot communicate.

$$\begin{aligned}
&(\mathrm{BCP}_\bot + \mathrm{RSE})(A) \vdash \\
&\quad P \owedge a \,.(\neg P \owedge 1) \parallel P :\!\rightarrow b.(\neg P :\!\rightarrow 1) \\
&= P \owedge a \,.(\neg P \owedge 1) \between P :\!\rightarrow b.(\neg P :\!\rightarrow 1) \\
&\qquad + P :\!\rightarrow b.(\neg P :\!\rightarrow 1) \between P \owedge a \,.(\neg P \owedge 1) \\
&\qquad + P \owedge a \,.(\neg P \owedge 1) \mid P :\!\rightarrow b.(\neg P :\!\rightarrow 1) \\
&= P \owedge (a.(\neg P \owedge 1) \between P :\!\rightarrow b.(\neg P :\!\rightarrow 1)) \\
&\qquad + P :\!\rightarrow (b.(\neg P :\!\rightarrow 1) \between P \owedge a \,.(\neg P \owedge 1)) \\
&\qquad + P \owedge 0 \\
&= P \owedge a \,.(\neg P \owedge 1 \parallel P :\!\rightarrow b.(\neg P :\!\rightarrow 1)) \\
&\qquad + P \owedge P :\!\rightarrow b \,.(\neg P :\!\rightarrow 1 \parallel P \owedge a \,.(\neg P \owedge 1)) \\
&= P \owedge a \,.(\neg P \owedge 0 + P :\!\rightarrow b.(\neg P :\!\rightarrow 1 \parallel \neg P \owedge 1) + \neg P \owedge 0) \\
&\qquad + P \owedge b \,.(\neg P :\!\rightarrow (1 \between P \owedge a \,.(\neg P \owedge 1)) \\
&\qquad\qquad + P \owedge a \,.(\neg P \owedge 1 \parallel \neg P :\!\rightarrow 1) + P \owedge 0) \\
&= P \owedge a \,.(\neg P \owedge P :\!\rightarrow b \,.(\neg P :\!\rightarrow 1 \parallel \neg P \owedge 1)) \\
&\qquad + P \owedge b \,.(\neg P :\!\rightarrow 0 \\
&\qquad\qquad + P \owedge a \,.(\neg P \owedge 0 + \neg P :\!\rightarrow 0 + \neg P \owedge \neg P :\!\rightarrow 1)) \\
&= P \owedge a \,.(\neg P \owedge 0) + b.(P \owedge a \,.(\neg P \owedge 1)).
\end{aligned}$$

https://doi.org/10.1017/CBO9781139195003.014 Published online by Cambridge University Press

Note that this example illustrates that signal observation provides an *asynchronous* communication and synchronization mechanism, as opposed to the *synchronous* mechanism provided via the communication merge and communication function.

Theory $(\mathrm{BCP}_\bot + \mathrm{RSE})(A, \gamma)$ admits an elimination result. Closed terms can be rewritten into (basic) $(\mathrm{BSP}_\bot + \mathrm{RSE})(A)$-terms. The ground-extension is furthermore conservative.

Using the elimination result, it is possible to determine the root signal of arbitrary closed $(\mathrm{BCP}_\bot + \mathrm{RSE})(A, \gamma)$-terms. The root signal of a basic term as defined in Definition 10.4.2 (Basic $(\mathrm{BSP}_\bot + \mathrm{RSE})(A)$-terms) is the signal χ in the format given in that definition. Applying the root-signal operator to such a basic term results in term $\chi \mathbin{\triangle} 0$. Exercise 10.4.8 illustrates several identities that can be derived for the root-signal operator. As a note aside, it is interesting to observe that there is no such simple identity for sequential composition (when that operator is added to the theory), see Exercise 10.4.10. The root signal emitted by a sequential composition is the signal emitted by the first operand of the composition, unless this term can terminate, in which case the root signal of the composition is the conjunct of the signals emitted by the two operands.

Some of the additional operational rules underlying the term model for process theory $(\mathrm{BCP}_\bot + \mathrm{RSE})(A, \gamma)$ are presented in Table 10.13, namely the rules for the merge and root-signal operators. Note that only one rule is necessary for the root-signal operator, which defines the consistency predicate. The absence of any other rules illustrates that a 'root-signal process' cannot perform any actions nor terminate successfully.

Example 10.4.7 (Signal observation: a traffic light) A more realistic example of signal observation than the example given in Example 10.4.6 is a traffic light, or, more precisely, the interaction between a car driver and a traffic light. Assume a set of propositional variables $\{red, yellow, green\}$. The following specification defines a (Dutch) traffic light, as it is observed by a passing car driver. The latter explains the termination options in the specification, signifying the option to stop observing the traffic light.

$$\begin{aligned} TL_r &= (red \wedge \neg yellow \wedge \neg green) \mathbin{\triangle} (1 + change.TL_g), \\ TL_y &= (\neg red \wedge yellow \wedge \neg green) \mathbin{\triangle} (1 + change.TL_r), \\ TL_g &= (\neg red \wedge \neg yellow \wedge green) \mathbin{\triangle} (1 + change.TL_y)\ . \end{aligned}$$

The following specifies a careful car driver that (initially) stops not only at a red light, but also at a yellow light.

$$\frac{\_\_\ TDS((\mathrm{BCP}_\perp + \mathrm{RSE})(A,\gamma), \mathit{effect})\ \_\_\_\_\_}{TDS((\mathrm{BSP}_\perp + \mathrm{RSE})(A), \mathit{effect});}$$

unary: rs, $(\partial_H)_{H \subseteq A}$; binary: $\_ \parallel \_$, $\_ \between \_$, $\_ \mid \_$;

x, x', y, y';

$$\frac{\langle x, v\rangle \xrightarrow{a} \langle x', v'\rangle \quad \langle y, v\rangle \searrow \quad \langle y, v'\rangle \searrow}{\langle x \parallel y, v\rangle \xrightarrow{a} \langle x' \parallel y, v'\rangle}$$

$$\frac{\langle x, v\rangle \xrightarrow{a} \langle x', v'\rangle \quad \langle y, v\rangle \xrightarrow{b} \langle y', v''\rangle \quad \gamma(a,b) = c \quad \langle x' \parallel y', \mathit{effect}(c, v)\rangle \searrow}{\langle x \parallel y, v\rangle \xrightarrow{c} \langle x' \parallel y', \mathit{effect}(c, v)\rangle}$$

$$\frac{\langle x, v\rangle\downarrow \quad \langle y, v\rangle\downarrow}{\langle x \parallel y, v\rangle\downarrow} \qquad \frac{\langle x, v\rangle \searrow \quad \langle y, v\rangle \searrow}{\langle x \parallel y, v\rangle \searrow}$$

$$\ldots$$

$$\frac{\langle x, v\rangle \searrow}{\langle rs(x), v\rangle \searrow}$$

$$\ldots$$

Table 10.13. Term deduction system for $(\mathrm{BCP}_\perp + \mathrm{RSE})(A, \gamma)$ (with $a, b, c \in A$, $v, v', v'' \in BV$).

$$CD = \mathit{approach}.(\mathit{green} :\rightarrow \mathit{drive}.1 + \neg \mathit{green} :\rightarrow \mathit{stop}.(\mathit{green} :\rightarrow \mathit{start}.(\neg \mathit{red} :\rightarrow \mathit{drive}.1))).$$

Expression $TL_r \parallel CD$ now describes the expected interaction between an initially red traffic light and an approaching driver. There is no communication. The transition system is shown in Figure 10.3. States are aligned in rows, according to the color of the traffic light. The colors are of course red, yellow, and green, reading from top to bottom. Signals are not shown. Action *change* changes the values of the propositional values as expected. Other actions do not affect these values.

Example 10.4.8 (Communication and signal observation) For a slightly more elaborate example, that illustrates the trade-off between (synchronous) communication and (asynchronous) signal observation, reconsider the communicating buffers of Section 7.6, illustrated in Figure 7.3. The original specification describes two one-place buffers communicating over ports i and l, and l and o, respectively. The parallel composition gives a two-place buffer

https://doi.org/10.1017/CBO9781139195003.014 Published online by Cambridge University Press

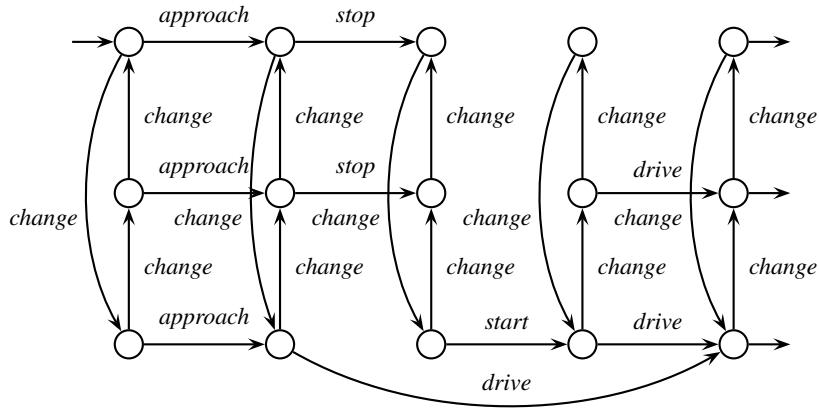


Fig. 10.3. The interaction between a traffic light and a car driver.

with ports i and o, when enforcing communication over the internal port l. The following specification adds signals that show the contents of the two one-place buffers at their respective output ports.

$$BS_{il} = show_l\emptyset \,\triangle\!\!\blacktriangle (1 + \sum_{d\in D} i?d.(show_l d \,\triangle\!\!\blacktriangle l!d.BS_{il})),$$
$$BS_{lo} = show_o\emptyset \,\triangle\!\!\blacktriangle (1 + \sum_{d\in D} l?d.(show_o d \,\triangle\!\!\blacktriangle o!d.BS_{lo})) .$$

It is now possible to repeat the derivations for the two-place buffer of Section 7.6, with the only difference that the content of the buffers is visible over ports l and o.

Signals as used in the above specification allow moving the parameter d in the actions and communications to a signal observation. Define the action set as $\{p?, p!, p⁉ \mid p \in \{i, l, o\}\}$. Assume there is only one communication, namely $\gamma(l?, l!) = l⁉$. Consider the following specification.

$$BS2_{il} = show_l\emptyset \,\triangle\!\!\blacktriangle (1 + \sum_{d\in D} show_i d :\to i?.(show_l d \,\triangle\!\!\blacktriangle l! .BS2_{il})),$$
$$BS2_{lo} = show_o\emptyset \,\triangle\!\!\blacktriangle (1 + \sum_{d\in D} show_l d :\to l?.(show_o d \,\triangle\!\!\blacktriangle o! .BS2_{lo})) .$$

Assume furthermore that all of the basic signals are exclusive, i.e., the following signal is an invariant in the above processes:

$$\bigwedge_{p\in\{i,l,o\}}((show_p\emptyset \supset \bigwedge_{d\in D} \neg show_p d) \wedge \bigwedge_{d\in D}(show_p d \supset (\neg show_p\emptyset \wedge \bigwedge_{e\in D\setminus\{d\}} \neg show_p e))).$$

Note that this can be achieved by adding the signal as a conjunct to all of the signals shown in the specifications. The invariant is omitted from the specifications for readability.

https://doi.org/10.1017/CBO9781139195003.014 Published online by Cambridge University Press

The process $\partial_H(BS2_{il} \,\|\, BS2_{lo})$, with $H = \{l?, l!\}$, now again specifies a two-place buffer. Note that this two-place buffer can only perform input actions under the condition that $show_i d$ is satisfied. This condition makes explicit that the environment is expected to provide data to the buffer.

In order to get rid of the last remaining communication, intuitively two extra signals per port are needed, because the original communication is synchronous, whereas signal observations are asynchronous. The *rdy* (ready) signals in the following specifications indicate that a buffer is ready to receive a value over its input port; the *flg* (flag) signals indicate that a buffer is ready to provide an output over the indicated port. The specifications use actions *in* and *out* for obtaining input and providing output. These actions modify the ready and flag signals as appropriate. Reset actions are included in the specifications to reset signals to their original settings after an output has been acknowledged by the environment. None of the actions can communicate. Observe that the specifications do not allow simultaneous inputs and outputs. Specification $BS3_{lo}$, for example, requires $\neg flg_l$ to hold before it can perform an out_o.

$$
\begin{aligned}
BS3_{il} = {} & (rdy_i \wedge show_l\emptyset \wedge \neg flg_l) \,{}^{\wedge\!\blacktriangle}(\\
& 1 + \textstyle\sum_{d\in D} (flg_i \wedge show_i d) :\!\rightarrow in_i \,. \\
& \qquad (\neg rdy_i \wedge show_l d \wedge flg_l) \,{}^{\wedge\!\blacktriangle}(rdy_l \wedge \neg flg_i) :\!\rightarrow out_l \,. \\
& \qquad (\neg rdy_i \wedge show_l d \wedge flg_l) \,{}^{\wedge\!\blacktriangle}(\neg rdy_l :\!\rightarrow reset_l.BS3_{il})), \\
BS3_{lo} = {} & (rdy_l \wedge show_o\emptyset \wedge \neg flg_o) \,{}^{\wedge\!\blacktriangle}(\\
& 1 + \textstyle\sum_{d\in D} (flg_l \wedge show_l d) :\!\rightarrow in_l \,. \\
& \qquad (\neg rdy_l \wedge show_o d \wedge flg_o) \,{}^{\wedge\!\blacktriangle}(rdy_o \wedge \neg flg_l) :\!\rightarrow out_o \,. \\
& \qquad (\neg rdy_l \wedge show_o d \wedge flg_o) \,{}^{\wedge\!\blacktriangle}(\neg rdy_o :\!\rightarrow reset_o.BS3_{lo})).
\end{aligned}
$$

Assuming as before the invariant mentioned above, it can again be shown that the parallel composition behaves as a two-place buffer.

As a final observation, note that the flag and ready signals in the above specifications are technically redundant. It is possible to encode the synchronization behavior with the existing *show* signals; see Exercise 10.4.14.

Observe from the various examples given so far that synchronization between signals is enforced via the signal names. In a more general setting, one could consider a signal communication function that relates signals that are expected to match. In the buffers example, for example, it would then be possible to connect buffers with arbitrary ports, stating that port p is connected to port q for arbitrary p and q. Furthermore, note that in the above buffers example, communications over internal ports are visible. To abstract from those internal actions, the present theory needs to be extended with abstraction as considered

in Chapter 8. Both extensions do not present problems, and are not treated here.

It is also desirable to have a means to abstract from signals at internal ports, to hide those typically meaningless signals from an external observer. This section is concluded by describing a mechanism for signal hiding. Notation $P\Delta x$ denotes that propositional variable P is hidden in process expression x.

Table 10.14 gives axioms for signal hiding, leading to theory $(\mathrm{BSP}_\bot + \mathrm{SHD})(A)$. The essential idea is that a propositional variable that is hidden is replaced by both true and false, which are the two values that it can possibly take. Axioms SHD2–5 follow the structure of basic terms (see Definition 10.4.2), with additional signal emissions in Axioms SHD2–4, for which the reason is explained below. Axiom SHD1 states the effect of signal hiding on the inaccessible process. Since $\bot$ is derivably equal to $\mathit{false} \triangleright 0$, Axiom SHD1 is derivable from the other axioms of the theory using Axiom SHD2.

The crucial axiom in theory $(\mathrm{BSP}_\bot + \mathrm{SHD})(A)$ is Axiom SHD5. It states that signal hiding distributes over choice when preserving the root signal of the two alternatives. This is in line with the fact that signals emitted by alternatives of a process carry over to the process as a whole. Axiom SHD5 is the reason that it is necessary to include the root-signal operator of theory $(\mathrm{BCP}_\bot + \mathrm{RSE})(A, \gamma)$ in the current theory, with the same axioms. The presence of the root-signal terms in Axiom SHD5 is also the reason why it is necessary to explicitly add signals in the left-hand sides of Axioms SHD2–4. Without those signals, it would not be possible to eliminate signal-hiding operators from (closed) terms involving choices.

As a final remark, note that after an action is executed, both options to replace a propositional variable by true or false must be included again, as signals are not persistent.

The following is a simple example of a calculation.

$$\begin{aligned}(\mathrm{BSP}_\bot + \mathrm{SHD})(A) &\vdash P\Delta(a.(P \triangleright 1) + a.(\neg P \triangleright 1))\\ &= a.(\mathit{true} \triangleright \mathit{true} :\to 1) + a.(\mathit{true} \triangleright \mathit{true} :\to 1) = a.1.\end{aligned}$$

The term deduction system in Table 10.15 reflects that the value of a propositional variable in a state does not matter when it is hidden.

Exercises

10.4.1 Show that Axioms IP1, $x + \bot = \bot$, and RSE3, $\phi \triangleright \bot = \bot$, are derivable from the other axioms of theory $(\mathrm{BSP}_\bot + \mathrm{RSE})(A)$.

10.4.2 Show that the following identities are derivable from theory $(\mathrm{BSP}_\bot + \mathrm{RSE})(A)$:

https://doi.org/10.1017/CBO9781139195003.014 Published online by Cambridge University Press

$\_\_(\mathrm{BSP}_\bot + \mathrm{SHD})(A)\_\_\_\_\_$
$(\mathrm{BSP}_\bot + \mathrm{RSE})(A)$;

unary: rs, $(P\Delta\_)_{P\in\mathbf{P}}$;

x, y;

$P\Delta\bot = \bot$	SHD1
$P\Delta(\phi \trianglerighteq 0) = (\phi[true/P] \vee \phi[false/P]) \trianglerighteq 0$	SHD2
$P\Delta(\phi \trianglerighteq \psi :\to 1) = (\phi[true/P] \vee \phi[false/P]) \trianglerighteq$ $((\phi \wedge \psi)[true/P] :\to 1 + (\phi \wedge \psi)[false/P] :\to 1)$	SHD3
$P\Delta(\phi \trianglerighteq \psi :\to a.x) = (\phi[true/P] \vee \phi[false/P]) \trianglerighteq$ $((\phi \wedge \psi)[true/P] :\to a.(P\Delta x)$ $+(\phi \wedge \psi)[false/P] :\to a.(P\Delta x))$	SHD4
$P\Delta(x + y) = P\Delta(rs(y) + x) + P\Delta(rs(x) + y)$	SHD5
$rs(\bot) = \bot$	RS1
$rs(1) = 0$	RS2
$rs(0) = 0$	RS3
$rs(a.x) = 0$	RS4
$rs(x + y) = rs(x) + rs(y)$	RS5
$rs(\phi :\to x) = \phi :\to rs(x)$	RS6
$rs(\phi \trianglerighteq x) = \phi \trianglerighteq rs(x)$	RS7

Table 10.14. The process theory BSP(A) with signal hiding (with $\phi, \psi \in \mathbf{FB}$ and $P \in \mathbf{P}$ a propositional variable).

(a) $\phi \trianglerighteq x = (\phi \trianglerighteq 0) + x$;

(b) $(\phi \trianglerighteq x) + (\psi \trianglerighteq y) = (\phi \wedge \psi) \trianglerighteq (x + y)$.

10.4.3 Prove Proposition 10.4.3 (Reduction to basic terms).

10.4.4 Prove that theory $(\mathrm{BSP}_\bot + \mathrm{RSE})(A)$ is a conservative ground-extension of theory $(\mathrm{BSP}_\bot + \mathrm{GC})(A)$.

10.4.5 Modify the specification of the stack in Example 10.4.5 (Stack) so that the top element is only shown when an action *top* is executed. Executing actions *top* and $pop(d)$ on an empty stack will cause an *error* signal to be emitted. Give different variants of the stack, where an error leads to deadlock, or to a state of underflow from which recovery is possible by execution of a push action.

10.4.6 Show that equation $\bot \parallel x = \bot$ is derivable from theory $(\mathrm{BCP}_\bot + \mathrm{RSE})(A, \gamma)$.

10.4.7 Show that equation $\phi \trianglerighteq x \| y = \phi \trianglerighteq (x \parallel y)$ is *not* derivable from theory $(\mathrm{BCP}_\bot + \mathrm{RSE})(A, \gamma)$.

10.4.8 Prove that, for any closed $(\mathrm{BCP}_\bot + \mathrm{RSE})(A, \gamma)$-terms p and q and any $H \subseteq A$, the following identities for the root-signal operator are derivable from theory $(\mathrm{BCP}_\bot + \mathrm{RSE})(A, \gamma)$:

https://doi.org/10.1017/CBO9781139195003.014 Published online by Cambridge University Press

$TDS((\mathrm{BSP}_\perp + \mathrm{SHD})(A), \mathit{effect})$
$TDS((\mathrm{BSP}_\perp + \mathrm{RSE})(A), \mathit{effect})$;

unary: rs, $(P\Delta\_)_{P\in\mathbf{P}}$;

x, x';

$$\frac{\langle x, v[\mathit{true}/P]\rangle \xrightarrow{a} \langle x', v'\rangle}{\langle P\Delta x, v\rangle \xrightarrow{a} \langle P\Delta x', \mathit{effect}(a, v)\rangle} \qquad \frac{\langle x, v[\mathit{false}/P]\rangle \xrightarrow{a} \langle x', v'\rangle}{\langle P\Delta x, v\rangle \xrightarrow{a} \langle P\Delta x', \mathit{effect}(a, v)\rangle}$$

$$\frac{\langle x, v[\mathit{true}/P]\rangle\downarrow}{\langle P\Delta x, v\rangle\downarrow} \qquad \frac{\langle x, v[\mathit{false}/P]\rangle\downarrow}{\langle P\Delta x, v\rangle\downarrow}$$

$$\frac{\langle x, v[\mathit{true}/P]\rangle \searrow}{\langle P\Delta x, v\rangle \searrow} \qquad \frac{\langle x, v[\mathit{false}/P]\rangle \searrow}{\langle P\Delta x, v\rangle \searrow}$$

$$\frac{\langle x, v\rangle \searrow}{\langle rs(x), v\rangle \searrow}$$

Table 10.15. Term deduction system for $(\mathrm{BSP}_\perp + \mathrm{SHD})(A)$ (with $a \in A$, $v, v' \in BV$, and $P \in \mathbf{P}$).

(a) $rs(p \mid q) = rs(q) + rs(q)$;
(b) $rs(p \between q) = rs(p)$;
(c) $rs(p \parallel q) = rs(p) + rs(q)$;
(d) $rs(\partial_H(p)) = rs(p)$.

10.4.9 Prove that, for any closed $(\mathrm{BCP}_\perp + \mathrm{RSE})(A, \gamma)$-term p, $(\mathrm{BCP}_\perp + \mathrm{RSE})(A, \gamma) \vdash rs(p) = \phi \mathbin{⦸} 0$, for some $\phi \in \mathbf{FB}$.

10.4.10 Show that both $rs(x \cdot y) = rs(x) + rs(y)$ and $rs(x \cdot y) = rs(x)$ are not derivable from theory $(\mathrm{TCP}_\perp + \mathrm{RSE})(A, \gamma)$ (i.e., $(\mathrm{BCP}_\perp + \mathrm{RSE})(A, \gamma)$ extended with sequential composition), not even for closed terms.

10.4.11 Give details for the construction of term models for the various equational theories introduced in this section. Investigate elimination, soundness, ground-completeness, and conservativity.

10.4.12 Derive a recursive specification in $(\mathrm{BCP}_\perp + \mathrm{RSE})(A, \gamma)$ for the process $TL_r \parallel CD$ of Example 10.4.7 (Signal observation: traffic light). Also make the *effect* function specifying the effect of actions on valuations of propositional variables explicit.

10.4.13 Derive recursive specifications in $(\mathrm{BCP}_\perp + \mathrm{RSE})(A, \gamma)$ for the three buffer processes $\partial_H(BS_{il} \parallel BS_{lo})$ (with $H = \{l?d, l!d \mid d \in D\}$), $\partial_H(BS2_{il} \parallel BS2_{lo})$ (with $H = \{l?, l!\}$) and $BS3_{il} \parallel BS3_{lo}$ of Example 10.4.8 (Communication and signal observation). Make the *effect* function specifying the effect of actions on valuations of propositional variables explicit, and draw a transition system for each of the three

https://doi.org/10.1017/CBO9781139195003.014 Published online by Cambridge University Press

processes. Finally, apply signal hiding, to hide all signals in the resulting specifications.

10.4.14 Give an alternative specification for the buffer processes $BS3_{il}$ and $BS3_{lo}$ of Example 10.4.8 (Communication and signal observation) that does not use communication or any of the flag or ready signals. That is, encode all the required synchronization in the *show* signals. Show that the alternative specifications when put in parallel yield the same result as $BS3_{il} \parallel BS3_{lo}$.

10.5 State operators

So far in this chapter, the state a process is in is not visible completely, is not directly accessible in the equational theory. The state is only visible through validity of propositional formulas, the emitted signals. In the operational semantics, the state is modeled by means of a valuation, that gives the value of all data variables. In this section, process states are modeled more explicitly in the equational theory, by means of a parameter attached to a unary operator, resulting in the class of so-called *state operators*.

First of all, suppose that a state space S is given. This set is left unspecified, but could consist of valuations of variables. Let us assume there is a special state $\perp \in S$ that denotes the inaccessible state. Further, there are three functions:

(i) A signal function $sig : S \to \mathbf{FB}$, that denotes the signal that is emitted in the state. Always, it is required that $sig(\perp) = false$. If it is not necessary to use signals in a particular setting, $sig(s) = true$ can be stipulated for all $s \in S \setminus \{\perp\}$.

(ii) A function $effect : A \times S \to S$, that denotes the resulting state if $a \in A$ is executed in state $s \in S$. If $effect(a, s) = \perp$, then a is blocked, cannot be executed in state s; $effect(a, \perp) = \perp$, for all $a \in A$.

(iii) A function $action : A \times S \to A$, which is a generalization of a renaming function. It denotes the action that can be observed if $a \in A$ is executed in state $s \in S$. Usually, this function can be taken to be trivial, i.e., $action(a, s) = a$ can be defined for all $a \in A, s \in S$.

In some applications, it is useful to use more than one state operator in a specification. For this reason, a state operator takes a second parameter. Process $\lambda_s^m(x)$ denotes that *machine* m executes program x and is currently in state s, where the machine m is a 4-tuple of the state set and the functions introduced above. That is, $m = (S, sig, effect, action)$. Let M be the set of all

https://doi.org/10.1017/CBO9781139195003.014 Published online by Cambridge University Press

machines (for the given action set A). The equational theory for state operators is presented in Table 10.16.

$(\mathrm{BSP}+\mathrm{SO})(A)$	
$(\mathrm{BSP}_\perp+\mathrm{RSE})(A)$;	
unary: $(\lambda_s^m)_{m=(S,sig,effect,action)\in M,s\in S}$;	
x, y;	
$\lambda_s^m(\perp)=\perp$	SO1
$\lambda_\perp^m(x)=\perp$	SO2
$\lambda_s^m(0)=sig(s)\mathbin{\triangle}0$	SO3
$\lambda_s^m(1)=sig(s)\mathbin{\triangle}1$	SO4
$\lambda_s^m(a.x)=sig(s)\mathbin{\triangle}action(a,s).\lambda_{effect(a,s)}^m(x)$	SO5
$\lambda_s^m(x+y)=\lambda_s^m(x)+\lambda_s^m(y)$	SO6
$\lambda_s^m(\phi:\!\to x)=sig(s)\mathbin{\triangle}\phi:\!\to\lambda_s^m(x)$	SO7
$\lambda_s^m(\phi\mathbin{\triangle}x)=\phi\mathbin{\triangle}\lambda_s^m(x)$	SO8

Table 10.16. The process theory $(\mathrm{BSP}+\mathrm{SO})(A)$ (with $a\in A, \phi\in\mathbf{FB}$ and $s\in S\setminus\{\perp\}$).

Operational rules are presented in Table 10.17. Observe that the *effect* function is no longer a parameter of the operational semantics, because it is part of the equational theory. The development of a term model for which the equational theory is sound and ground-complete goes as before. Also extensions go along the usual lines. The machine superscript is omitted when it is clear from the context.

$TDS((\mathrm{BSP}+\mathrm{SO})(A))$	
$TDS((\mathrm{BSP}_\perp+\mathrm{RSE})(A), effect)$;	
unary: $(\lambda_s^m)_{m=(S,sig,effect,action)\in M,s\in S}$;	
x, x';	

$$\frac{\langle sig(s)\mathbin{\triangle}x, v\rangle\xrightarrow{a}\langle x', v'\rangle \quad b=action(a,s)\quad s'=effect(a,s)\quad v'(sig(s'))=true}{\langle\lambda_s^m(x), v\rangle\xrightarrow{b}\langle\lambda_{s'}^m(x'), v'\rangle}$$

$$\frac{\langle sig(s)\mathbin{\triangle}x, v\rangle\downarrow}{\langle\lambda_s^m(x), v\rangle\downarrow}\qquad\frac{\langle sig(x)\mathbin{\triangle}x, v\rangle\not\searrow}{\langle\lambda_s^m(x), v\rangle\not\searrow}$$

Table 10.17. Term deduction system for $(\mathrm{BSP}+\mathrm{SO})(A)$ (with $a, b\in A$, $s'\in S$, $v, v'\in BV$).

https://doi.org/10.1017/CBO9781139195003.014 Published online by Cambridge University Press