6
Sequential processes

6.1 Sequential composition

In the process theory BSP(A) discussed in Chapter 4, the only way of combining two processes is by means of alternative composition. For the specification of more complex systems, additional composition mechanisms are useful. This chapter treats the extension with a *sequential-composition* operator. Given two process terms x and y, the term $x \cdot y$ denotes the sequential composition of x and y. The intuition of this operation is that upon the *successful* termination of process x, process y is started. If process x ends in a deadlock, also the sequential composition $x \cdot y$ deadlocks. Thus, a pre-requisite for a meaningful introduction of a sequential-composition operator is that successful and unsuccessful termination can be distinguished. As already explained in Chapter 4, this is not possible in the theory MPT(A) as all processes end in deadlock. Thus, as before, as a starting point the theory BSP(A) of Chapter 4 is used. This theory is extended with sequential composition to obtain the *Theory of Sequential Processes* TSP(A). It turns out that the empty process is an identity element for sequential composition: $x \cdot 1 = 1 \cdot x = x$.

6.2 The process theory TSP

This section introduces the process theory TSP, the Theory of Sequential Processes. The theory has, as before, a set of actions A as its parameter. The signature of the process theory TSP(A) is the signature of the process theory BSP(A) extended with the sequential-composition operator.

To obtain the axioms of TSP(A), the axioms from Table 6.1 are added to the axioms of the process theory BSP(A) from Table 4.3. Axiom A5 states that sequential composition is associative. As mentioned before, and now formally captured in Axioms A8 and A9, the empty process is an identity element with

https://doi.org/10.1017/CBO9781139195003.010 Published online by Cambridge University Press

respect to sequential composition. Axiom A7 states that after a deadlock has been reached no continuation is possible. Note that the theory does not contain the axiom $x \cdot 0 = 0$; so the inaction constant 0 is not a 'true' zero for sequential composition. Axiom A4 describes the distribution of sequential composition over alternative composition from the right. The other distributivity property is not desired as it does not respect the moment of choice (recall Example 4.2.2 (The lady or the tiger?)). Finally, Axiom A10 describes the relation between sequential composition and action prefixes. The unary action-prefix operators bind stronger than the binary sequential composition.

TSP(A)			
BSP(A)			
binary: $\cdot$;			
x, y, z;			
$(x + y) \cdot z = x \cdot z + y \cdot z$	A4	$0 \cdot x = 0$	A7
$(x \cdot y) \cdot z = x \cdot (y \cdot z)$	A5	$x \cdot 1 = x$	A8
$a.x \cdot y = a.(x \cdot y)$	A10	$1 \cdot x = x$	A9

Table 6.1. The process theory TSP(A) (with $a \in A$).

Axiom A5 (associativity of sequential composition) is redundant in the sense that for closed TSP(A)-terms, it can be proven from the other axioms (see Exercise 6.2.8). This means that the axiom is not necessary to obtain a ground-complete axiomatization of the standard term model for TSP(A). This could be a reason not to include the axiom in the theory. However, as soon as recursion comes into play, it is needed to allow all derivations of interest.

The addition of the sequential-composition operator to the process syntax has been motivated from a user's perspective. From the expressiveness point of view, with respect to the description of processes by means of closed process terms (see Definition 5.7.2 (Expressiveness)), the addition of this operator is not needed. This is shown in the following *elimination theorem*. It states that for any closed term described using sequential composition, there is also an equivalent description in which sequential composition is not used.

Theorem 6.2.1 (Elimination) For any closed TSP(A)-term p, there exists a closed BSP(A)-term q such that TSP(A) $\vdash p = q$.

Proof The property is proven by providing a term rewriting system with the same signature as TSP(A) such that

https://doi.org/10.1017/CBO9781139195003.010 Published online by Cambridge University Press

(i) each rewrite step transforms a process term into a process term that is derivably equal,
(ii) the term rewriting system is strongly normalizing, and
(iii) no closed normal form of the term rewriting system contains a sequential-composition operator.

Consider the term rewriting system that has the following rewrite rules, obtained directly from the axioms of TSP(A) (Table 6.1) by replacing $=$ by $\rightarrow$; for any $a \in A$, and TSP(A)-terms x, y, z:

$$(x + y) \cdot z \rightarrow x \cdot z + y \cdot z,$$
$$(x \cdot y) \cdot z \rightarrow x \cdot (y \cdot z),$$
$$0 \cdot x \rightarrow 0,$$
$$x \cdot 1 \rightarrow x,$$
$$1 \cdot x \rightarrow x,$$
$$a.x \cdot y \rightarrow a.(x \cdot y).$$

A consequence of the construction of the rules from the axioms of TSP(A) is that each rewrite step transforms a term into a term that is derivably equal.
The second step of the proof, the strong normalization of the term rewriting system, is left to the reader, as an exercise (Exercise 6.2.4).
The last part of the proof is to show that no closed normal form of the term rewriting system contains a sequential-composition operator. Thereto, let u be a normal form of the above term rewriting system. Suppose that u contains at least one sequential-composition operator. Then, u must contain a subterm of the form $v \cdot w$ for some closed TSP(A)-terms v and w. This subterm can always be chosen in such a way that v is a closed BSP(A)-term. It follows immediately from the structure of closed BSP(A)-terms that one of the above rewrite rules can be applied to $v \cdot w$. As a consequence, u is not a normal form. This contradiction implies that u must be a closed BSP(A)-term. □

Exercises

6.2.1 Give a derivably equivalent closed BSP(A)-term for each of the following TSP(A)-terms:

(a) $a.1 \cdot b.1$;
(b) $(a.1 + b.1) \cdot (a.1 + b.1)$;
(c) $(a.1 + b.1) \cdot (a.1 + b.1) \cdot (a.1 + b.1)$.

6.2.2 From Theorem 6.2.1 (Elimination), it follows immediately that for any two closed BSP(A)-terms p and q, there exists a closed BSP(A)-term r such that TSP(A) $\vdash p \cdot q = r$. Prove this result, without using the elimination theorem.

https://doi.org/10.1017/CBO9781139195003.010 Published online by Cambridge University Press

6.2.3 Define the n-fold sequential composition $\_^n$ as an operator in TSP(A), by induction on n, in a similar way as in Notation 4.6.6 (n-fold action prefix). Show that $(a.1)^n = a^n 1$ for any $a \in A$, where $a^n\_$ is the n-fold action prefix.

6.2.4 Prove that the term rewriting system used in the proof of the elimination theorem (Theorem 6.2.1) is strongly normalizing.
(Hint: try induction on the weighted number of symbols in a term, where symbols in the left operand of a sequential composition operator have a higher weight than other symbols.)

6.2.5 Prove Theorem 6.2.1 (Elimination) by induction on the structure of closed TSP(A)-term p.

6.2.6 Show that in TSP(A), the axiom $x + x = x$ (A3) is equivalent to the equation $1 + 1 = 1$. More precisely, show that $\mathrm{TSP}(A) \vdash 1 + 1 = 1$ and that $\mathrm{A1}, \mathrm{A2}, 1 + 1 = 1, \mathrm{A4} - \mathrm{A10} \vdash x + x = x$.

6.2.7 Show that in TSP(A) the axiom $x + 0 = x$ (A6) is equivalent to the equation $1 + 0 = 1$. More precisely, show that $\mathrm{TSP}(A) \vdash 1 + 0 = 1$ and that $\mathrm{A1} - \mathrm{A5}, 1 + 0 = 1, \mathrm{A7} - \mathrm{A10} \vdash x + 0 = x$.

6.2.8 Prove that, for closed TSP(A)-terms p, q, and r, the associativity of sequential composition, i.e., $(p \cdot q) \cdot r = p \cdot (q \cdot r)$, is derivable from the other axioms of TSP(A).

6.3 The term model

The term model for the process theory TSP(A) is obtained in a similar way as earlier term models have been obtained. Starting from the term algebra $\mathbb{P}(\mathrm{TSP}(A))$, first the term deduction system for TSP(A) is defined, and then the quotient algebra $\mathbb{P}(\mathrm{TSP}(A))_{/\leftrightarrows}$ is obtained. It is shown that the process theory TSP(A) is indeed a sound and complete axiomatization of the term model $\mathbb{P}(\mathrm{TSP}(A))_{/\leftrightarrows}$.

Definition 6.3.1 (Term algebra) The *term algebra* for theory TSP(A) is the algebra $\mathbb{P}(\mathrm{TSP}(A)) = (\mathcal{C}(\mathrm{TSP}(A)), +, \cdot, (a.\_)_{a\in A}, 0, 1)$.

The term algebra is not a model of the process theory TSP(A). The set of closed terms $\mathcal{C}(\mathrm{TSP}(A))$ is turned into a transition-system space by means of a term deduction system. The term deduction system for TSP(A), given in Table 6.2, is obtained by extending the term deduction system for BSP(A) of Table 4.4 with deduction rules for the sequential-composition operator.

The first of these rules describes that the sequential composition of two processes p and q in the transition-system space has an option to terminate if both

https://doi.org/10.1017/CBO9781139195003.010 Published online by Cambridge University Press

TDS(TSP(*A*))
TDS(BSP(*A*));
binary: $\cdot$;
x, x', y, y';

$$\frac{x\downarrow \quad y\downarrow}{(x \cdot y)\downarrow} \qquad \frac{x \xrightarrow{a} x'}{x \cdot y \xrightarrow{a} x' \cdot y} \qquad \frac{x\downarrow \quad y \xrightarrow{a} y'}{x \cdot y \xrightarrow{a} y'}$$

Table 6.2. Term deduction system for TSP(A) (with $a \in A$).

p and q have this option; the second deduction rule states that actions from p can be executed; and the third deduction rule explains that actions from q can be executed in case p has an option to terminate.

Proposition 6.3.2 (Congruence) Bisimilarity is a congruence on $\mathbb{P}(\mathrm{TSP}(A))$.

Proof The property follows immediately from the format of the deduction rules in Tables 4.2, 4.4, and 6.2 by application of Theorem 3.2.7 (Congruence theorem). □

Definition 6.3.3 (Term model of TSP(A)**)** The term model of theory TSP(A) is the quotient algebra $\mathbb{P}(\mathrm{TSP}(A))_{/\leftrightarrow}$, see Definition 2.3.18 (Quotient algebra), where $\mathbb{P}(\mathrm{TSP}(A))$ is the term algebra defined in Definition 6.3.1.

Theorem 6.3.4 (Soundness) The process theory TSP(A) is a sound axiomatization of the algebra $\mathbb{P}(\mathrm{TSP}(A))_{/\leftrightarrow}$, i.e., $\mathbb{P}(\mathrm{TSP}(A))_{/\leftrightarrow} \models \mathrm{TSP}(A)$.

Proof The proof of this statement follows the same lines as the proof of soundness of BSP(A) with respect to $\mathbb{P}(\mathrm{BSP}(A))_{/\leftrightarrow}$ (Theorem 4.4.7). It must be shown that, for each axiom $s = t$ of TSP(A), $\mathbb{P}(\mathrm{TSP}(A))_{/\leftrightarrow} \models s = t$. The axioms of BSP($A$) form a subset of the axioms of TSP(A) and no deduction rules have been added for the operators from the signature of BSP(A). Therefore, the validity proof of the BSP(A) axioms remains valid. Hence, only the validity of the additional axioms A4, A5, and A7–A10 has to be proven. This is left as an exercise (Exercise 6.3.3). □

The above theorem has two immediate corollaries.

Corollary 6.3.5 (Soundness) Let s and t be two TSP(A)-terms. If $\mathrm{TSP}(A) \vdash s = t$, then $\mathbb{P}(\mathrm{TSP}(A))_{/\leftrightarrow} \models s = t$.

https://doi.org/10.1017/CBO9781139195003.010 Published online by Cambridge University Press

Corollary 6.3.6 (Soundness) Let p and q be two closed TSP(A)-terms. If TSP(A) $\vdash p = q$, then $p \leftrightarrows q$.

Theorem 6.3.7 (Conservative ground-extension) Process theory TSP(A) is a conservative ground-extension of process theory BSP(A).

Proof See Exercise 6.3.2. □

Theory TSP(A) is a conservative ground-extension of BSP(A). This is visualized in Figure 6.1. Obviously, by Theorem 4.4.1, TSP(A) is also a conservative ground-extension of MPT(A).

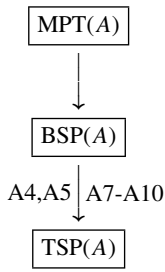


Fig. 6.1. TSP(A) is a conservative ground-extension of BSP(A).

Corollary 6.3.8 (Conservative ground-extension) Theory TSP(A) is a conservative ground-extension of theory MPT(A).

Theorem 6.3.9 (Ground-completeness) Process theory TSP(A) is a ground-complete axiomatization of the term model $\mathbb{P}(\text{TSP}(A))_{/\leftrightarrows}$, i.e., for any closed TSP(A)-terms p and q, $\mathbb{P}(\text{TSP}(A))_{/\leftrightarrows} \models p = q$ implies TSP(A) $\vdash p = q$.

Proof Using Theorem 3.2.19 (Operational conservativity), obviously *TDS*(TSP(A)) is an operational conservative extension of *TDS*(BSP(A)). As additionally BSP(A) is a ground-complete axiomatization of the model induced by *TDS*(BSP(A)) (Theorem 4.4.12), TSP(A) is a sound axiomatization of the model induced by *TDS*(TSP(A)) (Theorem 6.3.4), and TSP(A) has the elimination property with respect to BSP(A) (Theorem 6.2.1), from Theorem 3.2.26 (Ground-completeness), the desired result follows immediately. □

Corollary 6.3.10 (Ground-completeness) Let p and q be arbitrary closed TSP(A)-terms. If $p \leftrightarrows q$, then TSP(A) $\vdash p = q$.

https://doi.org/10.1017/CBO9781139195003.010 Published online by Cambridge University Press

Exercises

6.3.1 Draw the transition system of the following TSP(A)-terms:

(a) $(a.1 + 1) \cdot 0$;
(b) $(a.1 + 1) \cdot 1$;
(c) $(a.1 + 1) \cdot (a.1 + 1)$.

6.3.2 Prove Theorem 6.3.7 (Conservative ground-extension).

6.3.3 Finish the proof of soundness of theory TSP(A) for the term model $\mathbb{P}(\text{TSP}(A))_{/\leftrightarrow}$ (Theorem 6.3.4), i.e., prove that $\mathbb{P}(\text{TSP}(A))_{/\leftrightarrow} \models$ A4, A5, A7–A10.

6.4 Projection in TSP(A)

In Chapter 4, the process theory BSP(A) has been extended with projection operators to obtain (BSP + PR)(A). In this section, the extension of process theory TSP(A) along these lines is discussed. This extension of TSP(A) with the family of projection operators $(\pi_n)_{n\in\mathbb{N}}$ is called (TSP + PR)(A). The process theory (TSP + PR)(A) is obtained by extending the process theory TSP(A) with Axioms PR1 through PR5 of Table 4.5, or by extending (BSP + PR)(A) with the axioms of Table 6.1. In both cases, the same process theory results. Due to Theorem 6.2.1 (Elimination), no additional axioms are needed for the projection operators in combination with sequential composition. Given two closed TSP(A)-terms p and q, the projection $\pi_n(p \cdot q)$ can be computed by first reducing $p \cdot q$ to a closed BSP(A)-term and then applying the axioms for projection from the process theory (BSP + PR)(A). This observation implies several elimination results.

Theorem 6.4.1 (Elimination) For any closed (TSP + PR)(A)-term p, there exists a closed TSP(A)-term q such that TSP(A) $\vdash p = q$.

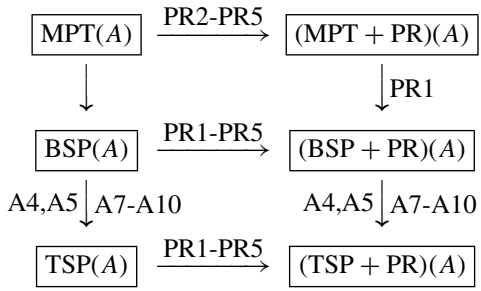
Proof Exercise 6.4.1 □

In addition, (TSP + PR)(A) has the elimination property with respect to theories BSP(A) and (BSP + PR)(A), i.e., closed (TSP + PR)(A)-terms can be reduced to BSP(A)- and (BSP + PR)(A)-terms, respectively.

The term deduction system underlying the standard term model of theory (TSP + PR)(A) consists of the deduction rules from the term deduction system for TSP(A) and the deduction rules from the term deduction system for (BSP + PR)(A). Furthermore, bisimilarity is a congruence on the term algebra of closed (TSP + PR)(A)-terms. The process theory (TSP + PR)(A) is a sound and ground-complete axiomatization of the corresponding term model

https://doi.org/10.1017/CBO9781139195003.010 Published online by Cambridge University Press

$\mathbb{P}((\mathrm{TSP}+\mathrm{PR})(A))_{/\leftrightarrow}$. Finally, $(\mathrm{TSP}+\mathrm{PR})(A)$ is a conservative ground-extension of both $\mathrm{TSP}(A)$ and $(\mathrm{BSP}+\mathrm{PR})(A)$, and hence also of theories $\mathrm{BSP}(A)$, $\mathrm{MPT}(A)$ and $(\mathrm{MPT}+\mathrm{PR})(A)$, as illustrated below.



Exercises

6.4.1 Prove Theorem 6.4.1 (Elimination). Thereto provide a term rewriting system such that

(a) each rewrite step transforms a process term into a process term that is derivably equal,
(b) the term rewriting system is strongly normalizing, and
(c) no closed normal form of the term rewriting system contains a projection operator.

(Hint: have a look at the proofs of Theorems 4.5.2 and 6.2.1.)

6.4.2 Prove by structural induction that for all closed $\mathrm{TSP}(A)$-terms p and q and $n \geq 0$,

$$(\mathrm{TSP}+\mathrm{PR})(A) \vdash \pi_n(p \cdot q) = \pi_n(\pi_n(p) \cdot \pi_n(q)).$$

6.5 Iteration

6.5.1 Prefix iteration

Another extension of basic process theories that has been discussed before is the introduction of a family of prefix-iteration operators $(a^*\_)_{a\in A}$. Analogous to the extension discussed in the previous section, it is possible to extend the process theory $\mathrm{TSP}(A)$ with these operators. The resulting theory is abbreviated $\mathrm{TSP}^*(A)$. The axioms of $\mathrm{TSP}^*(A)$ are the axioms of the process theory $\mathrm{TSP}(A)$, with additionally the prefix-iteration axioms PI1 and PI2 of Table 4.7 and Axiom PI3 given in Table 6.3. The unary prefix-iteration operators bind stronger than the binary sequential-composition operator.

https://doi.org/10.1017/CBO9781139195003.010 Published online by Cambridge University Press

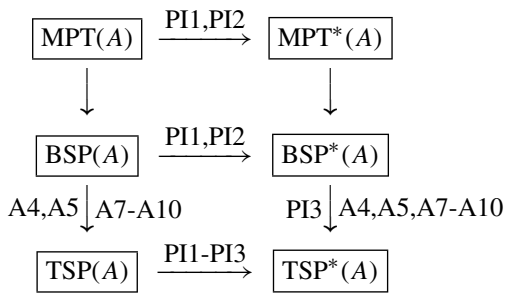
__ TSP*(A) _____
TSP(A), BSP*(A);
-
x, y;

$$a^*(x \cdot y) = a^*x \cdot y \qquad \text{PI3}$$

Table 6.3. The process theory TSP*(A) (with $a \in A$).

The process theory TSP*(A) does not have the elimination property for TSP(A), but it has the elimination property for BSP*(A).

The term deduction system underlying the term model for TSP*(A) consists of the rules of the term deduction systems for TSP(A) and BSP*(A). Bisimilarity is a congruence on the term algebra of closed TSP*(A)-terms. Theory TSP*(A) is a sound and ground-complete axiomatization of the term model $\mathbb{P}(\text{TSP}^*(A))_{/\leftrightarrows}$. TSP*(A) is a conservative ground-extension of both TSP(A) and BSP*(A).



6.5.2 General iteration

The sequential-composition operator can be considered as a generalization of the action-prefix operators, allowing to prefix a process with an arbitrary process. Similarly, the prefix-iteration operator can be generalized to a general *iteration operator*. Thus, the theory (TSP + IT)(A), the Theory of Sequential Processes with Iteration, is introduced. The signature of (TSP + IT)(A) is the signature of the process theory TSP(A) extended with the unary iteration operator _*. The axioms of (TSP + IT)(A) are obtained by adding the axioms of Table 6.4 to the axioms of the process theory TSP(A) of Table 6.1. As before, the unary iteration operator binds stronger than the binary sequential-composition operator.

Axiom IT1 can be compared to Axiom PI1 for prefix iteration: it states that x^* can execute its body at least once (followed by the iterated process

https://doi.org/10.1017/CBO9781139195003.010 Published online by Cambridge University Press

$(\mathrm{TSP}+\mathrm{IT})(A)$

$\mathrm{TSP}(A)$;

unary: $\_^*$;

x, y;

$x^* = x \cdot x^* + 1$	IT1
$(x+1)^* = x^*$	IT2
$(x+y)^* = x^* \cdot (y \cdot (x+y)^* + 1)$	IT3

Table 6.4. The process theory $(\mathrm{TSP}+\mathrm{IT})(A)$.

again), or terminate immediately. Axiom IT2 states that a 1 summand inside an iteration does not make a difference, as this is not something that can be executed and, by IT1, any iteration term can terminate already. Finally, Axiom IT3 is a difficult axiom due to Troeger (Troeger, 1993), that is not discussed here any further.

The axioms of $(\mathrm{TSP}+\mathrm{IT})(A)$ do not state anything explicitly about iterations of the basic inaction and empty processes. From IT1, however, it can easily be derived that $(\mathrm{TSP}+\mathrm{IT})(A) \vdash 0^* = 1$. Using IT2, it can then be inferred that $(\mathrm{TSP}+\mathrm{IT})(A) \vdash 1^* = 1$. Both results are as intuitively expected.

The iteration operator cannot be eliminated from closed terms in all circumstances, so there is no elimination theorem. This becomes more clear when the term model is considered.

A term deduction system for $(\mathrm{TSP}+\mathrm{IT})(A)$ is constructed by extending the term deduction system for $\mathrm{TSP}(A)$ of Table 6.2 with deduction rules for the iteration operator. Table 6.5 gives the term deduction system. The additional deduction rules should not come as a surprise.

$TDS((\mathrm{TSP}+\mathrm{IT})(A))$

$TDS(\mathrm{TSP}(A))$;

unary: $\_^*$;

x, x';

$$x^*\downarrow \qquad \frac{x \xrightarrow{a} x'}{x^* \xrightarrow{a} x' \cdot x^*}$$

Table 6.5. Term deduction system for $(\mathrm{TSP}+\mathrm{IT})(A)$ (with $a \in A$).

Based on the term deduction system, the term model $\mathbb{P}((\mathrm{TSP}+\mathrm{IT})(A))_{/\leftrightarrow}$ can be constructed, because, as before, bisimilarity is a congruence on the term algebra by the format of the deduction rules. Theory $(\mathrm{TSP}+\mathrm{IT})(A)$ is

https://doi.org/10.1017/CBO9781139195003.010 Published online by Cambridge University Press

a sound axiomatization of the term model, and the theory is a conservative ground-extension of $\mathrm{TSP}(A)$, but it is not a ground-complete axiomatization of the term model (see (Sewell, 1997)).

The theory $(\mathrm{TSP} + \mathrm{IT})(A)$ is more expressive than the theory $\mathrm{TSP}^*(A)$. In the term model of $\mathrm{TSP}^*(A)$, a loop can only occur from a certain state to itself. Or in other words, a loop can only consist of one action. On the other hand, in the term model of $(\mathrm{TSP} + \mathrm{IT})(A)$, processes can have loops consisting of multiple actions and states. Consider for example the process $(a.b.1)^*$, which has a cycle consisting of two states (see Exercise 6.5.2).

An interesting observation is that equational theory $(\mathrm{TSP} + \mathrm{IT})(A)$ can be viewed as a theory of regular expressions. When eliminating the action-prefix operators $a._-$ in favor of action constants $a.1$, $(\mathrm{TSP} + \mathrm{IT})(A)$ has the same constants and operators as the algebra of regular expressions known from automata theory (see e.g. (Linz, 2001)). The model of closed $(\mathrm{TSP} + \mathrm{IT})(A)$-terms modulo language equivalence known from automata theory (see Definition 3.1.7) is in fact the algebra of finite automata. In this book, theory $(\mathrm{TSP} + \mathrm{IT})(A)$, however, builds upon bisimulation equivalence instead of language equivalence. In automata theory, every finite automaton (regular process in the context of this book, see Section 5.8) can be expressed as a regular expression (closed $(\mathrm{TSP} + \mathrm{IT})(A)$-term). In the current context, it is not possible to represent all regular processes as closed $(\mathrm{TSP} + \mathrm{IT})(A)$-terms. Figure 6.2 shows a transition system for which it can be shown that there is no closed $(\mathrm{TSP} + \mathrm{IT})(A)$-term with a transition system in the term algebra $\mathbb{P}((\mathrm{TSP} + \mathrm{IT})(A))$ that is bisimilar to this transition system. The proof of this fact is beyond the scope of this text; the interested reader is referred to e.g. (Bergstra *et al.*, 2001). This example also shows that the theory $(\mathrm{TSP} + \mathrm{IT})(A)$ is less expressive than theory $\mathrm{BSP}_{\mathrm{gfrec}}(A)$ of Section 5.8. In other words, iteration is less expressive than general guarded finite recursion, which is not very surprising. It is interesting to observe that theories $(\mathrm{TSP} + \mathrm{IT})(A)$ and $\mathrm{BSP}_{\mathrm{gfrec}}(A)$ are equally expressive when language equivalence is the basic semantic equivalence. It follows from the results in Section 5.8 and language theory that both equational theories capture precisely all finite automata (regular transition systems in the terminology of this book).

Exercises

6.5.1 Prove that $(\mathrm{TSP} + \mathrm{IT})(A) \vdash 0^* = 1 = 1^*$.

6.5.2 Draw the transition system of the following $(\mathrm{TSP} + \mathrm{IT})(A)$-terms: $(a.b.1)^*$, $(a.0)^*$, $(a.b.1)^* \cdot 0$, $(a.1 + b.1)^*$.

https://doi.org/10.1017/CBO9781139195003.010 Published online by Cambridge University Press

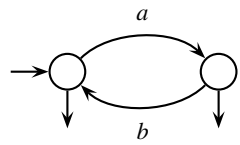


Fig. 6.2. A transition system that is not bisimilar to the transition system of any closed (TSP + IT)(A)-term.

6.6 Recursion

In Section 6.2, it has been shown that from each closed TSP(A)-term all occurrences of the sequential-composition operator can be eliminated (thus obtaining a closed BSP(A)-term). This, however, does not mean that the addition of sequential composition is without consequences. In a setting with recursion, the elimination property for sequential composition does not hold anymore. This section discusses the extension of TSP(A) with recursion.

6.6.1 The theory

The theory $\mathrm{TSP}_{\mathrm{rec}}(A)$, the Theory of Sequential Processes with Recursion, is obtained by extending the signature of theory TSP(A) with constants corresponding to all recursion variables in all recursive specifications of interest, and by adding all the recursive equations of these specifications as axioms to the theory, along the lines of Section 5.2. The term model $\mathbb{P}(\mathrm{TSP}_{\mathrm{rec}}(A))_{/\leftrightarrows}$ of $\mathrm{TSP}_{\mathrm{rec}}(A)$ is based on the term deduction system $TDS(\mathrm{TSP}_{\mathrm{rec}}(A))$, obtained by combining the term deduction systems of Tables 6.2 (Term deduction system of TSP(A)) and 5.2 (Term deduction system of $\mathrm{BSP}_{\mathrm{rec}}(A)$). It is straightforward to prove that $\mathrm{TSP}_{\mathrm{rec}}(A)$ is a sound axiomatization of $\mathbb{P}(\mathrm{TSP}_{\mathrm{rec}}(A))_{/\leftrightarrows}$.

There is no ground-completeness result, because in general the recursive equations do not capture all equalities between recursion variables that are valid in the term model. It is in general also not possible to eliminate recursion variables, and as a consequence, sequential-composition occurrences, from closed $\mathrm{TSP}_{\mathrm{rec}}(A)$-terms. However, the extension of TSP(A) with recursion is conservative, i.e., theory $\mathrm{TSP}_{\mathrm{rec}}(A)$ is a conservative ground-extension of process theory TSP(A). Intuitively, $\mathrm{TSP}_{\mathrm{rec}}(A)$ should also be a conservative ground-extension of $\mathrm{BSP}_{\mathrm{rec}}(A)$. However, none of the standard proof techniques, via term rewriting or the theory of structural operational semantics, can be applied to prove this result. Figure 6.3 shows the conservativity result and conjecture for theory $\mathrm{TSP}_{\mathrm{rec}}(A)$, extending Figures 5.4 and 6.1.

Recursion principles are needed to allow meaningful equational reasoning

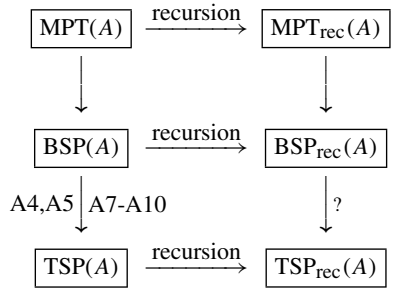


Fig. 6.3. Conservativity results for $\mathrm{TSP}_{\mathrm{rec}}(A)$.

in a context with recursion. Recall that principles AIP and AIP^- are only meaningful in a context with projection operators. Note that the definition of guardedness, Definition 5.5.8, carries over to $(\mathrm{TSP}+\mathrm{PR})_{\mathrm{rec}}(A)$-terms.

Theorem 6.6.1 (Recursion principles) Recursion principles RDP, RDP^-, RSP, and AIP^- are valid in the term model $\mathbb{P}((\mathrm{TSP}+\mathrm{PR})_{\mathrm{rec}}(A))_{/\leftrightarrows}$. Principle AIP is not valid in this model. RDP, RDP^-, and RSP are also valid in model $\mathbb{P}(\mathrm{TSP}_{\mathrm{rec}}(A))_{/\leftrightarrows}$.

The validity of principle RDP, and hence RDP^-, in both mentioned models follows immediately from the construction of the process theories and their term models. The proof of Theorem 5.5.23 (Validity of AIP^- in $\mathbb{P}((\mathrm{BSP}+\mathrm{PR})_{\mathrm{rec}}(A))_{/\leftrightarrows}$) carries over to the context of theory $(\mathrm{TSP}+\mathrm{PR})_{\mathrm{rec}}(A)$, showing the validity of principle AIP^- in $\mathbb{P}((\mathrm{TSP}+\mathrm{PR})_{\mathrm{rec}}(A))_{/\leftrightarrows}$. The invalidity of AIP in $\mathbb{P}((\mathrm{TSP}+\mathrm{PR})_{\mathrm{rec}}(A))_{/\leftrightarrows}$ follows from the same example as used to show the invalidity of AIP in model $\mathbb{P}((\mathrm{BSP}+\mathrm{PR})_{\mathrm{rec}}(A))_{/\leftrightarrows}$ (see Theorem 5.5.20). Assuming the validity of RSP in model $\mathbb{P}((\mathrm{TSP}+\mathrm{PR})_{\mathrm{rec}}(A))_{/\leftrightarrows}$, its validity in $\mathbb{P}(\mathrm{TSP}_{\mathrm{rec}}(A))_{/\leftrightarrows}$ follows from the fact that $(\mathrm{TSP}+\mathrm{PR})_{\mathrm{rec}}(A)$ is a ground-conservative extension of $\mathrm{TSP}_{\mathrm{rec}}(A)$, following the reasoning in the proof of Theorem 5.5.11 (Validity of RSP in $\mathrm{BSP}_{\mathrm{rec}}(A)$). Finally, the validity of RSP in $\mathbb{P}(\mathrm{TSP}_{\mathrm{rec}}(A))_{/\leftrightarrows}$ follows from Theorem 5.5.29 (Relation between the recursion principles) and the following proposition.

Proposition 6.6.2 (HNF property) Process theory $(\mathrm{TSP}+\mathrm{PR})_{\mathrm{rec}}(A)$ satisfies the HNF property, as defined in Definition 5.5.24 (Head normal form).

Proof The proof goes via induction on the structure of a completely guarded $(\mathrm{TSP}+\mathrm{PR})_{\mathrm{rec}}(A)$-term s, similar to the proof of Proposition 5.5.26 (HNF property for $(\mathrm{BSP}+\mathrm{PR})_{\mathrm{rec}}(A)$). Five out of six cases in the current

https://doi.org/10.1017/CBO9781139195003.010 Published online by Cambridge University Press

proof are straightforward adaptations of the corresponding cases in the proof of Proposition 5.5.26. The sixth case is the case that $s \equiv s' \cdot s''$, for two completely guarded $(\mathrm{TSP} + \mathrm{PR})_{\mathrm{rec}}(A)$-terms s' and s''. It then follows by induction that both s' and s'' can be rewritten into head normal forms, say t' and t'', respectively. Proposition 5.5.25 (Head normal forms) implies that there exists a natural number $n \in \mathbf{N}$ such that, for any $i < n$, there are $a_i \in A$ and $(\mathrm{TSP} + \mathrm{PR})_{\mathrm{rec}}(A)$-terms t_i such that

$$(\mathrm{TSP} + \mathrm{PR})_{\mathrm{rec}}(A) \vdash t' = \sum_{i<n} a_i.t_i(+1).$$

Then, applying the axioms of $(\mathrm{TSP} + \mathrm{PR})_{\mathrm{rec}}(A)$,

$$(\mathrm{TSP} + \mathrm{PR})_{\mathrm{rec}}(A) \vdash s = t' \cdot t'' = (\sum_{i<n} a_i.t_i(+1)) \cdot t'' = \sum_{i<n} (a_i.t_i) \cdot t''(+1 \cdot t'') = \sum_{i<n} a_i.(t_i \cdot t'')(+t'').$$

The last term in this derivation is a head normal form, which completes the proof. □

6.6.2 The stack revisited

In Section 5.6, a stack has been specified using $\mathrm{BSP}(A)$ with recursion. For this purpose, an infinite number of recursive equations was needed, even if the set D of data is finite. Using the additional syntax presented in this chapter, alternative specifications of a stack can be given and proven equal to the original one. These specifications are finite as long as the set D from which the elements in the stack are taken is finite, showing the usefulness of sequential composition.

Recall from Definition 5.5.19 (Generalized choice) that the use of the $\sum$-notation is only allowed for finite index sets. Using the syntax of $\mathrm{TSP}_{\mathrm{rec}}(A)$ and given a finite data set D, a recursive specification E for the stack can be given as follows:

$$Stack2 = 1 + \sum_{d \in D} push(d).Stack2 \cdot pop(d).Stack2.$$

Notice that this specification is guarded. In the equation, when an element d is pushed on the stack, a return is made to $Stack2$, which can push additional elements on the stack or terminate immediately. Eventually, d can be popped from the stack, and then the stack is empty, after which it can terminate or start over. Notice that, using the iteration operator of Section 6.5.2, this specification can also be given as follows:

$$Stack2 = (\sum_{d \in D} push(d).Stack2 \cdot pop(d).1)^*.$$

The two descriptions of the stack given in Section 5.6 and above are derivably equivalent.

https://doi.org/10.1017/CBO9781139195003.010 Published online by Cambridge University Press

Proposition 6.6.3 (Stacks) $(\mathrm{TSP}_{\mathrm{rec}} + \mathrm{RSP})(A) \vdash \mathit{Stack}1 = \mathit{Stack}2$.

To prove this proposition, introduce the following notation: for any $d \in D$ and arbitrary sequence $\sigma \in D^*$,

$$\begin{aligned} X_\epsilon &\equiv \mathit{Stack}2, \\ X_{d\sigma} &\equiv \mathit{Stack}2 \cdot \mathit{pop}(d).X_\sigma . \end{aligned}$$

Inspired by the transition systems of $\mathit{Stack}1$ and $\mathit{Stack}2$, it is proven that X_ϵ is a solution for $\mathit{Stack}1$ and X_σ is a solution for S_σ for all σ. Thereto, the following equations have to be proven:

$$\begin{aligned} &(\mathrm{TSP} + E)(A) \vdash X_\epsilon = X_\epsilon, \\ &(\mathrm{TSP} + E)(A) \vdash X_\epsilon = 1 + \sum_{d \in D} \mathit{push}(d).X_d, \\ &(\mathrm{TSP} + E)(A) \vdash X_{d\sigma} = \mathit{pop}(d).X_\sigma + \sum_{e \in D} \mathit{push}(e).X_{ed\sigma}. \end{aligned}$$

The first equation is trivial. The derivations for the second and third equation are as follows.

$$\begin{aligned} &(\mathrm{TSP} + E)(A) \vdash \\ &\quad X_\epsilon = \mathit{Stack}2 \\ &\qquad = 1 + \sum_{d \in D} \mathit{push}(d).\mathit{Stack}2 \cdot \mathit{pop}(d).\mathit{Stack}2 \\ &\qquad = 1 + \sum_{d \in D} \mathit{push}(d).(\mathit{Stack}2 \cdot \mathit{pop}(d).X_\epsilon) \\ &\qquad = 1 + \sum_{d \in D} \mathit{push}(d).X_d \end{aligned}$$

and

$$\begin{aligned} &(\mathrm{TSP} + E)(A) \vdash \\ &\quad X_{d\sigma} = \mathit{Stack}2 \cdot \mathit{pop}(d).X_\sigma \\ &\qquad = \left(1 + \sum_{e \in D} \mathit{push}(e).\mathit{Stack}2 \cdot \mathit{pop}(e).\mathit{Stack}2\right) \cdot \mathit{pop}(d).X_\sigma \\ &\qquad = 1 \cdot \mathit{pop}(d).X_\sigma + \\ &\qquad\qquad \left(\sum_{e \in D} \mathit{push}(e).\mathit{Stack}2 \cdot \mathit{pop}(e).\mathit{Stack}2\right) \cdot \mathit{pop}(d).X_\sigma \\ &\qquad = \mathit{pop}(d).X_\sigma + \\ &\qquad\qquad \sum_{e \in D} (\mathit{push}(e).\mathit{Stack}2 \cdot \mathit{pop}(e).\mathit{Stack}2) \cdot \mathit{pop}(d).X_\sigma \\ &\qquad = \mathit{pop}(d).X_\sigma + \\ &\qquad\qquad \sum_{e \in D} \mathit{push}(e).\mathit{Stack}2 \cdot (\mathit{pop}(e).\mathit{Stack}2 \cdot \mathit{pop}(d).X_\sigma) \\ &\qquad = \mathit{pop}(d).X_\sigma + \\ &\qquad\qquad \sum_{e \in D} \mathit{push}(e).\mathit{Stack}2 \cdot \mathit{pop}(e).(\mathit{Stack}2 \cdot \mathit{pop}(d).X_\sigma) \end{aligned}$$

https://doi.org/10.1017/CBO9781139195003.010 Published online by Cambridge University Press

$$
\begin{aligned}
&= pop(d).X_\sigma + \sum_{e\in D} push(e).Stack2 \cdot pop(e).X_{d\sigma} \\
&= pop(d).X_\sigma + \sum_{e\in D} push(e).(Stack2 \cdot pop(e).X_{d\sigma}) \\
&= pop(d).X_\sigma + \sum_{e\in D} push(e).X_{ed\sigma}.
\end{aligned}
$$

At this point, it has been shown that any solution for *Stack*2 is also a solution for *Stack*1. As both recursive specifications involved are guarded, using RSP, each of them has precisely one solution in the model $\mathbb{P}(\mathrm{TSP}_{\mathrm{rec}}(A))_{/\leftrightarrows}$. Consequently, $(\mathrm{TSP}_{\mathrm{rec}}+\mathrm{RSP})(A)\vdash Stack1 = Stack2$, proving the above proposition.

Observe that the above derivations use associativity of sequential composition on terms containing recursion variables. This cannot be avoided, showing that, in a context with recursion, associativity axiom A5 is meaningful, whereas it could be omitted from the basic theory $\mathrm{TSP}(A)$ without many consequences, as explained in Section 6.2.

6.6.3 Some expressiveness aspects

In Section 5.8, the theory $\mathrm{BSP}_{\mathrm{gfrec}}(A)$ was introduced, and it was shown that guarded finite recursion in the context of $\mathrm{BSP}(A)$ coincides with regular processes. The stack process is not finitely definable over $\mathrm{BSP}(A)$ (see Definition 5.7.5 (Definability)). It cannot be specified as a closed $\mathrm{BSP}_{\mathrm{gfrec}}(A)$-term, and it is not regular. The guarded finite recursive specification for *Stack*2 in this section therefore shows that TSP with guarded finite recursion is more expressive than $\mathrm{BSP}_{\mathrm{gfrec}}(A)$. In fact, the specification for *Stack*2 shows that finite guarded recursive specifications over $\mathrm{TSP}(A)$ allow the specification of non-regular processes, or, in other words, that non-regular processes are finitely definable over $\mathrm{TSP}(A)$. It is interesting to look a bit closer into these expressiveness aspects, particularly in relation with guardedness (which is required by the definition of definability).

Note that the general theories $\mathrm{BSP}_{\mathrm{rec}}(A)$ and $\mathrm{TSP}_{\mathrm{rec}}(A)$ allowing arbitrary recursion are equally expressive, as defined in Definition 5.7.2 (Expressiveness). Process theory $\mathrm{BSP}_{\mathrm{rec}}(A)$ is already sufficiently powerful to specify all countable computable processes, see Section 5.7. The extension with sequential composition does not affect the expressiveness of the theory, because of the fundamental restriction that computable recursive specifications can only specify countable processes. Recall furthermore Theorem 5.7.7 (Processes definable over $\mathrm{BSP}(A)$) that states that a process is definable over $\mathrm{BSP}(A)$ if and only if it is finitely branching. Also this result carries over to the context with sequential composition, i.e., a process is definable over $\mathrm{TSP}(A)$ if and only if it is finitely branching. However, when allowing unguarded recursive

https://doi.org/10.1017/CBO9781139195003.010 Published online by Cambridge University Press

specifications, already a finite specification is sufficient to specify an infinitely branching process.

Consider the recursive specification

$$\{X = X \cdot (a.1) + a.1\}.$$

Using the deduction rules for recursion constants of Table 5.2, for any natural number n, the following transition can be derived:

$$X \xrightarrow{a} (a.1)^n,$$

where $\_^n$ is the n-fold sequential composition of Exercise 6.2.3. This can be proven by induction on the natural number n as follows:

(i) $n = 0$. Since $a.1 \xrightarrow{a} 1$, also $X \cdot (a.1) + a.1 \xrightarrow{a} 1$ and $X \xrightarrow{a} 1 \equiv (a.1)^0$.
(ii) $n > 0$. By induction, $X \xrightarrow{a} (a.1)^{n-1}$. Then, $X \cdot (a.1) \xrightarrow{a} (a.1)^{n-1} \cdot (a.1)$, i.e., $X \cdot (a.1) \xrightarrow{a} (a.1)^n$. This means that $X \cdot (a.1) + a.1 \xrightarrow{a} (a.1)^n$ and therefore also $X \xrightarrow{a} (a.1)^n$.

As for different natural numbers m and n, process terms $(a.1)^m$ and $(a.1)^n$ are not bisimilar, it must be concluded that the transition system associated with X has an infinite number of different successor states for the state associated with X. Hence, the process specified by recursion variable X is infinitely branching, showing that finite unguarded recursion can describe infinitely branching processes. (Note that the recursive specification does not *define* the process that X specifies via the default interpretation in the term model; since the recursive specification is unguarded, it has many other solutions as well (see Exercise 6.6.8).)

Another interesting observation is based on the following specification taken from (Bosscher, 1997):

$$\{X = a.(Y \cdot a.1), Y = a.(Y \cdot Z) + a.1, Z = a.1 + 1\}.$$

This finite guarded recursive specification over the signature of TSP(A) *defines*, as argued above, a finitely branching process, but interestingly there is no bound on the number of outgoing transitions that any single state of the process might have; the process exhibits so-called *unbounded* branching. This can be verified by drawing the transition system for recursion variable X, see Exercise 6.6.6. Thus, this example shows that unbounded branching processes are finitely definable over TSP(A). As shown in (Bosscher, 1997), it is not possible to finitely define processes with unbounded branching over theory BSP(A).

Finally, it is interesting to consider the expressiveness results in this subsection in the context of automata theory. Observe that recursive specification $E = \{X = X \cdot (a.1) + a.1\}$ discussed above corresponds to a left-linear grammar in the automata-theoretic context. In that context, left-linear grammars,

https://doi.org/10.1017/CBO9781139195003.010 Published online by Cambridge University Press

right-linear grammars, finite automata, and regular languages are all equivalent notions. However, in the current context, the process specified by specification E is infinitely branching, and thus non-regular.

Exercises

6.6.1 Prove, using AIP^-, that any solution of $X = a.X \cdot b.1$ is also a solution of $X = a.X$.

6.6.2 Prove that $((\mathrm{TSP} + \mathrm{PR})_{\mathrm{rec}} + \mathrm{AIP}^-)(A) \vdash \mu X.\{X = a.X\} \cdot \mu Y.\{Y = b.Y\} = \mu X.\{X = a.X\}$.

6.6.3 Given the syntax of $\mathrm{TSP}_{\mathrm{rec}}(A)$ and a finite data set D, a recursive specification for the stack using three equations can be given as follows:

$$\begin{aligned} \mathit{Stack}3 &= 1 + T \cdot \mathit{Stack}3, \\ T &= \sum_{d \in D} \mathit{push}(d).(U \cdot \mathit{pop}(d).1), \\ U &= 1 + T \cdot U. \end{aligned}$$

In this set of equations, $\mathit{Stack}3$ is specified as an iteration of T, which represents a terminating stack. In the specification of T, $U \cdot \mathit{pop}(d)$ can be seen as a stack element d, where U may or may not put new elements on top of the element, and where $\mathit{pop}(d)$ removes the element from the stack. Show that this recursive specification is guarded. Prove that $(\mathrm{TSP}_{\mathrm{rec}} + \mathrm{RSP})(A) \vdash \mathit{Stack}2 = \mathit{Stack}3$.

6.6.4 Consider yet another recursive specification for the stack, consisting of $|D| + 2$ equations, for data set D:

$$\begin{aligned} \mathit{Stack}4 &= 1 + T \cdot \mathit{Stack}4, \\ T &= \sum_{d \in D} \mathit{push}(d).T_d, \quad \text{and, for all } d \in D, \\ T_d &= \mathit{pop}(d).1 + T \cdot T_d. \end{aligned}$$

Show that this recursive specification is guarded. Prove that $(\mathrm{TSP}_{\mathrm{rec}} + \mathrm{RSP})(A) \vdash \mathit{Stack}2 = \mathit{Stack}4$.

6.6.5 A specification of a counter can be obtained by considering a stack that can only contain elements from a data set with one element. When, additionally, the atomic actions representing putting this element on the stack and taking it from the stack are represented by *plus* and *minus*, respectively, the following specification can be derived from the specification for $\mathit{Stack}4$ of the previous exercise:

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

https://doi.org/10.1017/CBO9781139195003.010 Published online by Cambridge University Press

Give a transition system for the process *Counter*. Show that this counter specification is derivably equivalent to the counter specification of Exercises 5.6.3 and 5.6.4.

6.6.6 Consider the recursive specification $\{X = a.(Y \cdot a.1), Y = a.(Y \cdot Z) + a.1, Z = a.1 + 1\}$. Draw the transition system of X (and observe that it is a transition system with unbounded branching).

6.6.7 Find two non-bisimilar transition systems that are both solutions of the (unguarded!) recursive equation $X = X \cdot a.1 + X \cdot b.1$.

6.6.8 Consider again recursive specification $\{X = X \cdot a.1 + a.1\}$ of the last subsection. Give a different solution for this recursive specification than the one already given. Give a plausible argument why any solution of this recursive specification must be infinitely branching.

6.6.9 Prove or disprove that $\mathrm{TSP}_{\mathrm{rec}}(A)$ is a conservative ground-extension of $\mathrm{BSP}_{\mathrm{rec}}(A)$. (Please inform the authors of any solution to this exercise.)

6.7 Renaming, encapsulation, and skip operators

In Exercises 5.6.3, 5.6.4 and 6.6.5, a counter was considered that can also be specified as a stack over a trivial singleton data type. However, for the stack, different action names are used (*push*, *pop*) than for the counter (*plus*, *minus*). In the exercises, the names were just changed without any further consideration. Often, it is desirable to have such a renaming inside the theory, in order to be able to deal with this in a more formal and precise way. This is not difficult to do. This section briefly presents the extension of the basic theory $\mathrm{BSP}(A)$ with renaming and two related families of operators, namely encapsulation and skip operators. The extension of $\mathrm{TSP}(A)$ with renaming is left as Exercise 6.7.7.

Suppose f is a renaming function, i.e., f is a function on atomic actions, $f : A \to A$. Then, a renaming operator ρ_f can be defined on process terms. Table 6.6 shows the extension of $\mathrm{BSP}(A)$ with renaming operators.

It is straightforward to obtain an elimination theorem and a conservativity theorem. Also the operational rules are not difficult; see Table 6.7. Again, it is a useful exercise to show that the resulting term model admits a soundness and a ground-completeness theorem.

The remainder of this section covers two other operators that turn out to be very useful: an operator that can block a certain action, and an operator that can skip a certain action. The former can be considered a renaming into 0, the latter a renaming into 1. Suppose H is a set of atomic actions, i.e., $H \subseteq A$. The operator ∂_H blocks execution of actions from H, and leaves other actions

__ (BSP + RN)(A) __
BSP(A);
unary: $(\rho_f)_{f:A\to A}$;
x, y;

$\rho_f(1) = 1$	RN1
$\rho_f(0) = 0$	RN2
$\rho_f(a.x) = f(a).\rho_f(x)$	RN3
$\rho_f(x + y) = \rho_f(x) + \rho_f(y)$	RN4

Table 6.6. BSP(A) with renaming (with $a \in A$ and $f : A \to A$).

__ *TDS*((BSP + RN)(A)) __
TDS(BSP(A));
unary: $(\rho_f)_{f:A\to A}$;
x, x';

$$\frac{x\downarrow}{\rho_f(x)\downarrow} \qquad \frac{x \xrightarrow{a} x'}{\rho_f(x) \xrightarrow{f(a)} \rho_f(x')}$$

Table 6.7. Term deduction system for BSP(A) with renaming (with $a \in A$ and $f : A \to A$).

unchanged. For reasons that become clear in the next chapter, this operator is called the *encapsulation operator*. Axioms are presented in Table 6.8.

__ (BSP + DH)(A) __
BSP(A);
unary: $(\partial_H)_{H\subseteq A}$;
x, y;

$\partial_H(1) = 1$		D1
$\partial_H(0) = 0$		D2
$\partial_H(a.x) = 0$	if $a \in H$	D3
$\partial_H(a.x) = a.\partial_H(x)$	otherwise	D4
$\partial_H(x + y) = \partial_H(x) + \partial_H(y)$		D5

Table 6.8. BSP(A) plus encapsulation (with $a \in A$, $H \subseteq A$).

Example 6.7.1 (Encapsulation) As an example, consider, on the one hand, the behavior of a machine performing process $a.(b.1 + c.1)$, that, for some reason, is unable to execute the atomic action c. Then, after having performed a, it will continue with b, as it does not have any other choice available.

https://doi.org/10.1017/CBO9781139195003.010 Published online by Cambridge University Press

On the other hand, consider a similar machine performing $a.b.1 + a.c.1$. Assuming again that c cannot be executed, this second machine will perform an a-step and next it will either perform b, or it will be in the position that it can only continue with the blocked c, which means it will end in a deadlock.

Using the encapsulation operator, this example can be modeled by blocking c in the above two specifications; thus, $H = \{c\}$. As a consequence,

$$\begin{aligned}&(\mathrm{BSP} + \mathrm{DH})(A) \vdash\\ &\quad \partial_H(a.(b.1 + c.1)) = a.\partial_H(b.1 + c.1) = a.(\partial_H(b.1) + \partial_H(c.1))\\ &\qquad = a.(b.\partial_H(1) + 0) = a.b.1,\end{aligned}$$

and, on the other hand,

$$\begin{aligned}&(\mathrm{BSP} + \mathrm{DH})(A) \vdash\\ &\quad \partial_H(a.b.1 + a.c.1) = \partial_H(a.b.1) + \partial_H(a.c.1)\\ &\qquad = a.\partial_H(b.1) + a.\partial_H(c.1) = a.b.1 + a.0.\end{aligned}$$

Notice that process $a.b.1 + a.0$ has a deadlock, whereas $a.b.1$ does not (see Definition 4.4.14).

Next, suppose I is a set of atomic actions, $I \subseteq A$. The operator ε_I skips the execution of actions from I, and leaves other actions unchanged. It is therefore called a *skip* operator. Axioms are presented in Table 6.9.

$(\mathrm{BSP} + \mathrm{EI})(A)$		
$\mathrm{BSP}(A)$;		
unary: $(\varepsilon_I)_{I \subseteq A}$;		
x, y;		
$\varepsilon_I(1) = 1$		E1
$\varepsilon_I(0) = 0$		E2
$\varepsilon_I(a.x) = \varepsilon_I(x)$	if $a \in I$	E3
$\varepsilon_I(a.x) = a.\varepsilon_I(x)$	otherwise	E4
$\varepsilon_I(x + y) = \varepsilon_I(x) + \varepsilon_I(y)$		E5

Table 6.9. $\mathrm{BSP}(A)$ with skip operators (with $a \in A$, $I \subseteq A$).

Also for the extensions with encapsulation and skip operators standard results can be established. Without further comment, Table 6.10 presents the operational rules for encapsulation. The rules for the skip operators are given in Table 6.11. The two rules in the bottom row of the table may need some explanation. As usual, the steps and the termination option for term $\varepsilon_I(p)$ are derived from the steps and termination option of p. The bottom left rule states that $\varepsilon_I(p)$ may terminate if p can do an a-step that is skipped and p', the result of executing a in p, can terminate *after* skipping all actions in I in p'. The last

https://doi.org/10.1017/CBO9781139195003.010 Published online by Cambridge University Press

provision is necessary because skipping actions can create extra termination options. The bottom right rule states that $\varepsilon_I(p)$ may do a b-step if p can do some a-step that is skipped followed by a b-step that is not skipped. Thus, the rules look ahead at the behavior of the p process after skipping the a action to determine the behavior of $\varepsilon_I(p)$.

$$\frac{\begin{array}{l}\_\_\ TDS((\mathrm{BSP}+\mathrm{DH})(A))\ \_\_\_\_\_\\ TDS(\mathrm{BSP}(A));\\ \text{unary: } (\partial_H)_{H\subseteq A}\,;\\ x, x';\end{array}}{\dfrac{x\downarrow}{\partial_H(x)\downarrow} \qquad \dfrac{x \xrightarrow{a} x' \quad a \notin H}{\partial_H(x) \xrightarrow{a} \partial_H(x')}}$$

Table 6.10. Term deduction system for encapsulation (with $a \in A$, $H \subseteq A$).

$$\frac{\begin{array}{l}\_\_\ TDS((\mathrm{BSP}+\mathrm{EI})(A))\ \_\_\_\_\_\\ TDS(\mathrm{BSP}(A));\\ \text{unary: } (\varepsilon_I)_{I\subseteq A}\,;\\ x, x', y;\end{array}}{\dfrac{x\downarrow}{\varepsilon_I(x)\downarrow} \qquad \dfrac{x \xrightarrow{a} x' \quad a \notin I}{\varepsilon_I(x) \xrightarrow{a} \varepsilon_I(x')}}$$

$$\frac{x \xrightarrow{a} x' \quad a \in I \quad \varepsilon_I(x')\downarrow}{\varepsilon_I(x)\downarrow} \qquad \frac{x \xrightarrow{a} x' \quad a \in I \quad \varepsilon_I(x') \xrightarrow{b} y}{\varepsilon_I(x) \xrightarrow{b} y}$$

Table 6.11. The term deduction system for skip operators (with $a, b \in A$ and $I \subseteq A$).

As a final remark, the extensions of TSP(A) with encapsulation and skip operators, (TSP + DH)(A) and (TSP + EI)(A), respectively, are obtained by simply combining the signatures and axiom sets of TSP(A) with those of (BSP + DH)(A) and (BSP + EI)(A), respectively. All standard results can be obtained as expected.

Exercises

6.7.1 Generalizing Definition 4.4.14 (Deadlock in BSP(A)-terms), a closed (BSP + DH)(A)-term is deadlock free if and only if it is derivably equal to a closed BSP(A)-term without 0 occurrence.

https://doi.org/10.1017/CBO9781139195003.010 Published online by Cambridge University Press

Simplify $\partial_{\{a\}}(a.0 + b.1)$. Note that the process $a.0 + b.1$ has a deadlock, but $\partial_{\{a\}}(a.0+b.1)$ does not. Hence, encapsulation can cause the loss of deadlock possibilities. Conversely, it is easy to find a closed BSP(A)-term p such that p is without deadlock, whereas $\partial_{\{a\}}(p)$ is not. Give such an example.

6.7.2 Give an example of a closed BSP(A)-term p such that p has a deadlock and $\varepsilon_I(p)$ for some appropriate I does no longer have a deadlock (assuming a generalization of Definition 4.4.14 (Deadlock in BSP(A)-terms) to closed (BSP + EI)(A)-terms).

6.7.3 Consider the behavior of the vending machine of Exercise 5.2.2 under the assumption that due to some mechanical defect it is impossible to insert any 10 cent coins. Does this machine have any deadlock?

6.7.4 Let id be the identity function on A and let $\circ$ denote function composition. Show that for all closed BSP(A)-terms p and renaming functions $f, g : A \to A$,

(a) $(\mathrm{BSP} + \mathrm{RN})(A) \vdash \rho_{id}(p) = p$,
(b) $(\mathrm{BSP} + \mathrm{RN})(A) \vdash \rho_f(\rho_g(p)) = \rho_{f \circ g}(p)$.

6.7.5 Consider theory (BSP + EI)(A) with recursion.

(a) Given the equation $X = \varepsilon_{\{a\}}(a.X)$, show that in the term model $\mathbb{P}((\mathrm{BSP} + \mathrm{EI})_{\mathrm{rec}}(A))_{/\leftrightarrow} \models X = 0$.
(b) Find two different solutions of the equation $X = \varepsilon_{\{a\}}(a.X)$ in the term model of (BSP + EI)(A) with recursion.
(c) Find two different solutions of the equation $X = a.\varepsilon_{\{a\}}(X)$ in the term model of (BSP + EI)(A) with recursion.
(d) Can you formulate a notion of guardedness in the presence of skip operators?

6.7.6 Consider the extensions of theory TSP(A) with encapsulation and skip operators, (TSP+DH)(A) and (TSP+EI)(A). Show by structural induction that for all closed TSP(A)-terms p, q and $H, I \subseteq A$,

(a) $(\mathrm{TSP} + \mathrm{DH})(A) \vdash \partial_H(p \cdot q) = \partial_H(p) \cdot \partial_H(q)$,
(b) $(\mathrm{TSP} + \mathrm{EI})(A) \vdash \varepsilon_I(p \cdot q) = \varepsilon_I(p) \cdot \varepsilon_I(q)$.

6.7.7 Develop the theory (TSP + RN)(A), TSP with renaming, i.e., the equational theory obtained by extending (BSP + RN)(A) with the sequential-composition operator and the axioms of Table 6.1. Prove elimination and conservativity results, and give a term model proving soundness and ground-completeness.

https://doi.org/10.1017/CBO9781139195003.010 Published online by Cambridge University Press

6.8 Bibliographical remarks

Taking the equational theory TSP(A), replacing processes $a.1$ by constants a, and subsequently removing action prefixing and the constants 0 and 1, results in the theory BPA(A) of (Bergstra & Klop, 1984a). The present treatment is from (Baeten, 2003), which in turn is based on (Koymans & Vrancken, 1985; Vrancken, 1997). The term deduction system of theory TSP(A) is due to (Baeten & Van Glabbeek, 1987).

Section 6.5 is based on (Bergstra *et al.*, 2001). Further work can be found in (Baeten *et al.*, 2007).

The specification of the stack in one equation is new here. The specification of the stack in three equations of Exercise 6.6.3 is due to (Koymans & Vrancken, 1985). The (earlier) one in $|D| + 2$ equations, where D is the data set, given in Exercise 6.6.4 is from (Bergstra & Klop, 1986a). The specification in one equation is possible due to the termination option of the empty stack. The specifications of (Koymans & Vrancken, 1985) and (Bergstra & Klop, 1986a) do not have such an initial termination option, and the specifications given in the current chapter have been adapted to include this termination option.

Section 6.7 is based on (Baeten & Bergstra, 1988). Renaming operators occur in most concurrency theories, see e.g. (Milner, 1980; Hoare, 1985). Encapsulation, and the notation ∂_H, is from (Bergstra & Klop, 1984a). For skipping, see (Baeten & Weijland, 1990).