Axiomatizing GSOS with termination

J.C.M. Baeten [a] & E.P. de Vink [a,b,1]

[a] *Department of Mathematics and Computer Science, Eindhoven University of Technology, P.O. Box 513, 5600 MB Eindhoven, The Netherlands*

[b] *LIACS, Universiteit Leiden, P.O. Box 9512, 2300 RA Leiden, The Netherlands*

Abstract

We discuss a combination of *GSOS*-type structural operational semantics with explicit termination, that we call the *tagh*-format (*tagh* being short for termination and *GSOS* hybrid). The *tagh*-format distinguishes between transition and termination rules, but, besides active and negative premises as in *GSOS*, also allows for what we call terminating and passive arguments. We extend the result of Aceto, Bloom and Vaandrager on the automatic generation of sound and complete axiomatizations for *GSOS* to the setting of *tagh*-transition systems. The construction of the equational theory is based upon the notion of a smooth and distinctive operation, which have been generalized from *GSOS* to *tagh*. We prove the soundness of the synthesized laws and show their completeness modulo bisimulation. The examples provided indicate a significant, though yet not ideal, improvement over the axiomatization techniques known so far.

Key words: Structured operational semantics, GSOS format, equational theories, bisimulation equivalence, complete axiomatizations, approximation induction principle

1 Introduction

It has become very popular in the concurrency community to define various process operators by means of Plotkin-style operational rules (see, for example, [AFV01]). These are usually pretty intuitive, and they can be used to derive a transition system for each process expression. Properties of such a transition system can then be checked using a model checker.

[1] Corresponding author: `evink@win.tue.nl`

But it is also well-known that this approach has its restrictions. Often, transition systems become too large to be handled by model checkers, or, due to the presence of parameters, transition systems may have infinitely many states. In these cases, an approach using theorem provers or employing equational reasoning can be very helpful.

In the face of these alternative approaches, it is often profitable to generate a set of laws or equations for an operator that is given by a set of operational rules. Moreover, we want two characterizations that match: the axiomatization should be sound and complete for the model of transition systems modulo (strong) bisimulation. The paper [ABV94] points the way in such an endeavour: in some cases an axiomatization can be derived by just following a recipe. Some other papers in this area are [Uli95,Uli00] (where other equivalence relations besides bisimulation equivalence are considered). However, in the years since the appearance of these papers, there have not been many applications of the theory. The reader may wonder why this is so.

In our opinion, this is due to the limited process algebraic basis employed in [ABV94]; in particular, termination and deadlock are identified. Any language involving parallel composition, be it a programming language or specification language, will experience the situation when no further action is possible, but components are not finished, e.g. when two components are waiting for different communications. This situation is usually called deadlock or unsuccessful termination. Now if the language also involves some form of sequential composition, we have to know when the first component in a sequential composition is finished, i.e. successfully terminated, in order for the second component to continue. In such a case, deadlock must be distinguished from successful termination, and, consequently, the axiomatization method of [ABV94] does not apply.

There are three ways to handle this combination of parallel composition and sequential composition. First, we can do away with sequential composition as a basic operator, only have prefixing as a rudimentary form of sequential composition, and use tricks like a special communication to mimic some form of sequential composition. This is the solution of CCS [Mil80,Mil89], and is, in our opinion, unsatisfactory. Second, we can use *implicit termination* as in ACP [BK84,BW90], where successful termination is implicitly attached to the last action. Finally, in the majority of cases, we find *explicit termination*, usually implemented by having two separate constants, one denoting deadlock, inaction or unsuccessful termination, the other one denoting skip or successful termination. Examples are LOTOS [Bri89], SDL [EHS97], CSP [BHR84], χ [BR97], and DiCons [BBM01]. See also [AH92]. The paper [Bae03] demonstrates how to restate ACP in terms of explicit termination and shows that this has advantages in extensions with timing. The book [BBR03] is a complete reworking of [BW90] taking explicit termination as a basic ingredient. Operationally, deadlock has no rules, and termination is denoted by a predicate on states.

In this paper, we adapt the theory of [ABV94] for the case of explicit termination. We think that the theory presented in this study can be extended in order to deal also with implicit termination, but leave this as future research. Starting from the *GSOS*-format (see [BIM95]), we extend it with termination to obtain the *tagh*-format (abbreviating 'termination and *GSOS* hybrid'). We also employ some additional generalizations so that auxiliary operators are needed in fewer cases: for instance, the definition of sequential composition does not require auxiliary operators as in [ABV94]. This does make the theory a lot more complicated, but we gain that the generated axiomatizations are closer to the ones in the literature, intuitively understandable, and are sound and complete for the model of transition systems modulo bisimulation.

The outcome is a recipe that can be applied in a straightforward manner. It is presented in Section 3. We also provide a few examples (sequential composition, leftmerge, disrupt and the priority operator) to illustrate the technique. Section 2 provides the necessary preliminaries, while Section 4 and Section 5 are devoted to the soundness and completeness of the generated theory. Some concluding remarks are collected in Section 6. We hope that our generalizations will lead to actual applications.

An extended abstract of this paper appeared in the Proceedings of STACS 2002 (see [BV02]). Full proofs are provided here. We thank the anonymous referees for their many insightful remarks and suggestions for improvements. These have led to many changes, e.g. in describing the role of passive and negative premises in operational rules.

2 Preliminaries

We assume the reader to be familiar with the standard notions and examples of process algebra (see, e.g., [Mil89,BW90,Fok00,BBR03]). Below we start out by presenting our semantic framework, a transition system space, where each state represents a transition system with termination. We continue with a basic process language with explicit termination, deadlock, prefixing and choice. We show how a transition system specification defines a transition system for each closed term over this signature.

Definition 1 A *transition-system space* over a set of labels L is a set S of *states*, equipped with one ternary relation $\rightarrow$ and one subset $\downarrow$:

(1) $\rightarrow \subseteq S \times L \times S$ is the set of *transitions*;

(2) $\downarrow \subseteq S$ is the set of *terminating* or *final* states.

The notation $s \xrightarrow{a} t$ is used for $(s, a, t) \in \rightarrow$, with the intuitive meaning that from s, there is an option to execute a and thereby proceed to state t, and $s\downarrow$ for $s \in \downarrow$, with the intuitive meaning that from s, there is an option to terminate immediately. The fact that for all states $t \in S$, $s \xrightarrow{a} t$ does not hold, is abbreviated as $s \overset{a}{\nrightarrow}$.

For each state $s \in S$, the *transition system* of s consists of all states reachable from s, and has the transitions and final states induced by the transition-system space. State s is called the *initial state* or *root* of the transition system of s.

Definition 2 A binary relation R on the set of states S of a transition-system space is a *bisimulation* relation if and only if the following so-called *transfer conditions* hold:

(1) 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 is a state $t' \in S$ such that $t \xrightarrow{a} t'$ and $(s', t') \in R$;
(2) 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 is a state $s' \in S$ such that $s \xrightarrow{a} s'$ and $(s', t') \in R$;
(3) whenever $(s, t) \in R$ and $s\downarrow$ then $t\downarrow$;
(4) whenever $(s, t) \in R$ and $t\downarrow$ then $s\downarrow$.

Two transition systems of $s, t \in S$ are *bisimulation equivalent* or *bisimilar*, notation $s \sim t$, if and only if there is a bisimulation relation R on S with $(s, t) \in R$.

It is a well-known fact that bisimilarity is an equivalence relation on the set of states of a transition system space. As a consequence, we can look at the quotient of S with respect to $\sim$, viz. the set of equivalence classes of S modulo $\sim$.

When proving soundness of the various laws that will be introduced in the sequel, the following property comes in handy. Its proof is immediate.

Lemma 3 Let t_1, t_2 be two states such that $t_1 \xrightarrow{a} t \iff t_2 \xrightarrow{a} t$ for all labels a and states t, and $t_1\downarrow \iff t_2\downarrow$. Then it holds that $t_1 \sim t_2$. $\square$

Next, we consider the following process signature. This language is called the language of Basic Sequential Processes (BSP) in [BBR03]. It has the following constants and operators, and is parametrised by an alphabet of actions Act. We will always assume this alphabet is finite. This makes it easier to reason about the finite projections of infinite processes in section 5.

(1) a constant ε, denoting immediate termination;

(2) a constant δ, denoting inaction or deadlock;
(3) a unary operation '$a.$ ' for every action a from Act, denoting prefixing of action a, and
(4) a binary operator '+', denoting (non-deterministic) choice.

Now, for the set of states S we will use the set of closed terms over this language, and for the set of labels L we will use the set of actions Act. We define a transition system space for a process language by means of *structured operational* rules, in a so-called *transition system specification*.

In the next section we shall be precise in what is a (tagh) transition system (see Definition 11). For the moment it suffices that a transition system specification is a finite collection of transition rules and termination rules. A transition rule and a termination rule are deduction rules of the form

$$t \xrightarrow{a} t' \quad \text{or} \quad \frac{s \xrightarrow{b} s'}{t \xrightarrow{a} t'}, \quad \text{and,} \quad t{\downarrow} \quad \text{or} \quad \frac{s{\downarrow}}{t{\downarrow}},$$

respectively, where t, t', s, s' and a, b are (possibly open) terms and actions of the relevant process language. The transition relation '$\rightarrow$' is subsequently defined as the least subset of $S \times L \times S$ containing all closed instances of transition rules of the form $t \xrightarrow{a} t'$ and closed under closed instances of transition rules of the form $\frac{s \xrightarrow{b} s'}{t \xrightarrow{a} t'}$. Likewise the termination predicate '$\downarrow$' is the smallest subset of S containing all closed instances of termination rules $t{\downarrow}$ and closed under closed instances of termination rules of the form $\frac{s{\downarrow}}{t{\downarrow}}$. The next definition provides a transition system specification for the Basic Sequential Processes introduced above.

Definition 4 The transition system specification TSS_0 for the process language BSP consists of the following rules.

$$a.x \xrightarrow{a} x \qquad \frac{x \xrightarrow{a} x'}{x + y \xrightarrow{a} x'} \qquad \frac{y \xrightarrow{a} y'}{x + y \xrightarrow{a} y'}$$

$$\varepsilon{\downarrow} \qquad \frac{x{\downarrow}}{(x + y){\downarrow}} \qquad \frac{y{\downarrow}}{(x + y){\downarrow}}$$

By means of these rules, we can generate a transition system space as follows. The two rules on the left are *axioms*: they state that for each closed term x, the state corresponding to term $a.x$ has a transition labeled with a to state x, and that the state corresponding to term ε is terminating. The other rules have hypotheses, so for instance the second rule says that whenever there is a transition labeled a from term x to term x', then there is also a transition labeled a from term $x + y$ to x'. By repeatedly applying the rules, the space is generated. In this simple case, it can be seen that the transition system of each closed term will be a finite graph with termination.

Later on, we will also use *negative premises* (premises of the form $s \stackrel{a}{\nrightarrow}$) in rules in a transition system specification. Then, it is not immediate to determine a transition system space generated by the rules. However, the restricted format that we will use ensures that a unique transition system space is generated, nonetheless.

As an alternative to its transition system, we can also provide an *equational theory* for this process language. Consider the following set of equations.

Definition 5 The equational theory ET_0 consists of the following equations:

$$\begin{aligned} x + y &= y + x \\ (x + y) + z &= x + (y + x) \\ x + x &= x \\ x + \delta &= x \end{aligned}$$

These equations induce a notion of equality on terms, by means of the well-known notion of *derivability* ($\vdash$). An equation can be derived from an equational theory by using substitution (any term can be substituted for a variable), contexts (if an equation is derivable, it can be applied in a context) and the equivalence properties of equality. The next basic soundness and completeness result can be shown with standard techniques. See, for example, [Mil89,BV95,BBR03]. A full proof can be found in [BBR03].

Theorem 6 *The equational theory ET_0 as given in Definition 5 is sound and complete for TSS_0 modulo bisimulation, i.e. each equation between two closed terms is derivable from the equational theory if and only if the corresponding transition systems induced by the transition system specification TSS_0 are bisimilar.* □

This presents the basic system. Over the years, very many extensions of this basic theory have been presented. The extended theories have greater expressiveness and improve readability of specifications. This paper makes it easy to generate, on the basis of operational rules, a set of axioms for such extensions so that soundness and completeness hold. Further on, in order to deal with negative premises in operational rules, we need one additional operator, viz. the unary one-step restriction operations ∂^1_B for every subset $B \subseteq Act$. The expression $\partial^1_B(t)$ indicates that the term t is not permitted to perform any action from B as a first step. However, this restriction is dropped after t has done a step outside of the action set B. The operator was called dagger in [ABV94].

The operational rules for this operator are presented in Definition 7a. With these rules added to the ones presented previously, we can generate a transition system for each closed term over the extended signature.

Definition 7

(a) The transition system specification TSS_∂ extends TSS_0 with the following transition and termination rules:

$$\frac{x \xrightarrow{a} x'}{\partial_B^1(x) \xrightarrow{a} x'}\ (a \notin B) \qquad \frac{x\downarrow}{\partial_B^1(x)\downarrow}$$

(b) The equational theory ET_∂ extends ET_0 with the following equations:

$$\begin{aligned}
\partial_B^1(x+y) &= \partial_B^1(x) + \partial_B^1(y) \\
\partial_B^1(a.x) &= a.x \ \text{ if } a \notin B \\
\partial_B^1(a.x) &= \delta \ \text{ if } a \in B \\
\partial_B^1(\delta) &= \delta \\
\partial_B^1(\varepsilon) &= \varepsilon
\end{aligned}$$

On the basis of the work of this paper, we can easily derive the axiomatization presented in Table 7b. The theorems presented further on imply that this axiomatization is sound and complete (a result that can also easily be proven directly). The first axiom is an example of a distributive law, the following two are action laws, then comes a deadlock law and finally we have a termination law.

Theorem 8 *The equational theory ET_∂ as given in Definition 7b is sound and complete for the transition system space generated by TSS_∂ modulo bisimulation.* □

This theorem follows from Sections 4 and 5 of this paper. The theory applies, as the operational rules added for the ∂_B^1 operators make them smooth and distinctive operators (see Definition 16). If there are no negative premises present, the ∂_B^1 operators are not needed in the axiomatization. The theorem can also be proven directly, using standard techniques as above. See also [ABV94].

In Section 5, on completeness, we make use of the concept of head normalization. In the context of process algebra with explicit termination its definition is as follows. We abbreviate a finite sum $t_{i_1} + \ldots + t_{i_n}$ by $\sum_{i \in I} t_i$ (where $I = \{i_1, \ldots, i_n\}$). $\sum_{i \in \emptyset} t_i$ is defined to be δ.

Definition 9 A term t of the form ε, δ, $\sum_{i \in I} a_i.t'_i$ or $(\sum_{i \in I} a_i.t'_i) + \varepsilon$ with I a finite non-empty index set, is in *head normal form*. An equational theory ET is *head normalizing* if for all terms t there exists a term t' in head normal form such that $ET \vdash t = t'$.

The notation $t_1 \equiv t_2$ is used to indicate the syntactic equality of the terms t_1 and t_2. We also use expressions like $C[x_k, y_\ell, z_m]$ to indicate that only variables from the set

$$\{ x_k \mid k \in K \} \cup \{ y_\ell \mid \ell \in L \} \cup \{ z_m \mid m \in M \}$$

occur in the context $C[\ ]$ with respect to some given (finite) index sets K, L and M.

Below we introduce various ways to add operators to a given transition system specification. The extra rules that are introduced should not affect the bisimilarity of terms of the earlier transition system. This is guaranteed when the extension is a so-called disjoint one.

Definition 10 A transition system specification TSS' is said to disjointly extend a transition system specification TSS if the rules in TSS' for the operators of TSS are exactly the same as the rules in TSS.

So, a disjoint extension TSS' does not introduce new transitions or termination for terms of TSS. Consequently, for terms t_1, t_2 of TSS, it holds that $t_1 \sim t_2 \iff t_1 \sim' t_2$, where '$\sim$' and '$\sim'$' denote bisimilarity of the transition systems of t_1 and t_2 with respect to TSS and TSS', respectively.

3 Generating equations for the *tagh*-format

In this section we introduce the *tagh*-format for transition system specifications. The acronym *tagh* stands for *termination and GSOS hybrid*. It extends the *GSOS*-format as introduced in [BIM95] with a notion of explicit termination. We provide, at the end of this section, a general procedure to obtain, for each transition system specification in *tagh*-format, a disjoint extension TSS' and an equational theory ET'. In later sections we prove the soundness and completeness of ET' for bisimulation on the transition system space generated by TSS'. As this transition system space extends the transition system space generated by TSS this amounts for terms t_1, t_2 over TSS to coincidence of bisimulation with respect to *TSS* and equality based on ET'. Thus, ET' is a sound and complete axiomatization of bisimulation on the transition system space generated by *TSS*.

Definition 11

(a) A *tagh*-transition rule ρ for an n-ary operation f is a deduction rule of the format below.

$$\frac{\{\, x_i \xrightarrow{a_{ip}} y_{ip} \mid i \in I, p \in P_i \,\} \quad \{\, x_j \stackrel{b}{\not\to} \mid j \in J, b \in B_j \,\} \quad \{\, x_k{\downarrow} \mid k \in K \,\}}{f(x_1, \ldots, x_n) \xrightarrow{a} C[x_m, y_{ip}]} \tag{1}$$

Here, $I, J, K \subseteq \{1, \ldots, n\}$, so these are subsets of the argument positions of f. Moreover, for each $i \in I$, we have that P_i is a nonempty finite index set; for each $j \in J$, we have that $B_j \subseteq Act$ is a (finite) set of actions (note that it is allowed that $B_j = \emptyset$); and, x_m, y_{ip}, for $m \in \{1, \ldots, n\}, i \in I, p \in P_i$ are pairwise distinct variables. These variables are the only variables that may occur in the context $C[x_m, y_{ip}]$.

(b) A *tagh*-termination rule θ for an n-ary operation f is a deduction rule of the format

$$\frac{\{\, x_k{\downarrow} \mid k \in K \,\}}{f(x_1, \ldots, x_n){\downarrow}} \tag{2}$$

with $x_1, \ldots, x_n$ pairwise distinct variables and the index set K any subset of the argument positions $\{1, \ldots, n\}$.

(c) Fix some finite collection of operations. A *tagh*-transition system specification is a finite transition system specification where any operation f different from 'ε', 'δ', '$a.$ ' for $a \in Act$, '$+$' and '∂^1_B' has transition rules and termination rules of the *tagh*-format only. The constants 'ϵ', 'δ' and operations '$a.$ ' for $a \in Act$. '$+$' and '∂^1_B' come equipped with the *tagh*-transition and *tagh*-termination rules as given in Section 2.

For notational convenience ρ with or without subscripts and superscripts is used to denote a transition rule, whereas θ, possibly decorated, is reserved for termination rules. In the context of a transition rule ρ of the format (1) we use $act(\rho)$ for the index set I of active variables. Likewise $term(\rho)$ denotes the index set K of terminated variables. Each negative premise actually occurring in a rule leads to a negative variable position. However, a variable x_j not occurring in a negative premise can still be taken to be negative by taking the set B_j to be empty (and vice versa). Thus, $neg(\rho)$, the index set J of negative variables, is not uniquely determined by a given rule. All variables that are not active, terminated nor negative are called passive. Thus, $pass(\rho)$, the index set of passive variables, is given by $L = \{1, \ldots, n\} \setminus (I \cup J \cup K)$. For a termination rule θ conforming to equation (2) we put $term(\theta) = K$. For a transition rule ρ like (1), we refer to $f(x_1, \ldots, x_n)$, or an instantiation of it, as the source of ρ, and to the term $C[x_m, y_{ip}]$ as its target. Occasionally we will write $t\not\downarrow$ if *not* $t\downarrow$, i.e., t cannot terminate immediately.

For a *tagh*-transition system specification, it can be seen that it generates a unique transition system space on closed terms. For, the transition systems of constants are given by axioms (0 argument positions), next, the transition system is determined for each term of the form $f(t_1, \ldots, t_n)$ where each t_i is a constant. Continuing in this fashion, the space is built up following the complexity of terms. Therefore, a straightforward induction on the complexity of terms shows that the resulting transition system space is finitely branching (this is because the set of actions *Act* is finite).

The *tagh*-format is an extension of the *GSOS*-format of [BIM95]. If we strip all aspects of termination from the definition we end up with the original format for *GSOS*. We have, as the *tagh*-format is subsumed by the *panth*-format of [Ver95], that bisimulation is a congruence, just as for *GSOS*. The syntactic format of general *tagh*-transition rules though, is much too liberal to allow for an automatic generation of axioms directly. Following [ABV94], we therefore introduce a more restricted format of transition rules, called smooth rules, where there are no clashes between active, negative, terminating and passive arguments. Also an active position is not permitted to have multiple transitions, i.e. each index set P_i is a singleton. In addition, there are restrictions regarding the variables that are allowed to occur in the target of the rule. For the synthesis of equational laws characterizing an operation f it is profitable to further restrict the collection of rules. In essence we want that at any time at most one of the transition rules for f applies. If the rules for f have this additional property, the operation is called smooth and distinctive.

Definition 12 Let *TSS* be a *tagh*-transition system specification. A transition rule ρ in *TSS* for an n-ary operation f is *smooth* if it is of the following format (3) and satisfies the following restrictions.

$$\frac{\{\, x_i \xrightarrow{a_i} y_i \mid i \in I \,\} \quad \{\, x_j \stackrel{b}{\not\rightarrow} \mid j \in J, b \in B_j \,\} \quad \{\, x_k\downarrow \mid k \in K \,\}}{f(x_1, \ldots, x_n) \xrightarrow{a} C[y_i, x_j, x_\ell\,]} \tag{3}$$

where

- the index sets I, J, K, L are pairwise disjoint and cover the set $\{\, 1, \ldots, n \,\}$,
- the index set I is non-empty, if $n > 0$,
- for $j \in J$, $B_j \subseteq Act$ a (necessarily finite but possibly empty) subset of actions
- in the target $C[y_i, x_j, x_\ell\,]$ only variables from the variable set $\{\, y_i \mid i \in I \,\} \cup \{\, x_p \mid p \in J \cup L \,\}$ occur.

The operation f is *smooth* with respect to *TSS* if all of its transition rules in *TSS* are smooth, and, moreover,

- for each position p in $\{1, \ldots, n\}$ it holds that $p \notin pass(\rho)$ for some rule ρ for f in *TSS*.

If an operation f in a *tagh*-transition system specification is smooth, there is often still a choice whether to call a certain position a passive position, or a negative position with empty set B_j. This freedom of choice can be exploited when we want to order the rules of f.

Example 13 Let the operator f be given by the following transition system specification:

$$\frac{x \xrightarrow{a} x'}{f(x,y,z) \xrightarrow{a} A} \ (1) \qquad\qquad \frac{x \xrightarrow{c} x'}{f(x,y,z) \xrightarrow{c} C} \ (3)$$

$$\frac{y \xrightarrow{b} y' \ x{\downarrow}}{f(x,y,z) \xrightarrow{b} B} \ (2) \qquad\qquad \frac{z \xrightarrow{d} z' \ x{\downarrow}}{f(x,y,z) \xrightarrow{d} D} \ (4)$$

For rule (1) the variables y and z can both be regarded either as passive or negative; the variable x is clearly active. So, we can interpret rule (1) as having index sets $I = \{x\}$, $J = \emptyset$, $K = \emptyset$, $L = \{y, z\}$ with both y and z as passive variables, but also with $I = \{x\}$, $J = \{y\}$, $K = \emptyset$, $L = \{z\}$ and $I = \{x\}$, $J = \{x\}$, $K = \emptyset$, $L = \{y\}$ with only one of y and z passive, or even with $I = \{x\}$, $J = \{y, z\}$, $K = \emptyset$, $L = \emptyset$ with both y and z regarded as negative variables. The same variation in classification of variables applies to rule (3). As rules (2) and (4) rate the variables y and z as active, we clearly have that this operator f is smooth.

We proceed to define an ordering on the rules of a smooth operation in a *tagh*-transition system specification. In general, this will only be a partial ordering.

Definition 14 Let *TSS* be a *tagh*-transition system specification. Let f be a smooth operation in *TSS*. Define sets I, J, K, L as given in (3). Then we say a transition rule ρ has a ranking $\langle L, I, K, J\rangle$. We define a partial ordering on rankings as follows: $\langle L, I, K, J\rangle \succcurlyeq \langle L', I', K', J'\rangle$ iff

- $J = J'$, $L \supseteq L'$ and $K \subseteq K'$, and
- $L \neq L' \implies I \cap K' \neq \emptyset$.

The ordering '$\succcurlyeq$' is referred to as a ranking of f.

Recall that a given transition rule can have more than one ranking, because of the undeterminateness of negative variables. As the presentation of transition rules have usually the ranking implicit instead of organizing the premises in sets as in (1) one has to decide what ranking applies.

Lemma 15 Let the operation f be smooth. Then the relation $\succcurlyeq$ on the rules of f is a partial ordering.

Proof First of all, reflexivity is immediate.

For anti-symmetry, suppose $\langle L, I, K, J\rangle \succcurlyeq \langle L', I', K', J'\rangle$ and $\langle L', I', K', J'\rangle \succcurlyeq \langle L, I, K, J\rangle$. From the first condition, we obtain $J = J'$, $L = L'$, $K = K'$, and since the remaining positions must be the active positions, also $I = I'$. This means $\langle L, I, K, J\rangle = \langle L', I', K', J'\rangle$. The second condition never applies.

Transitivity: if $\langle L, I, K, J\rangle \succcurlyeq \langle L', I', K', J'\rangle \succcurlyeq \langle L'', I'', K'', J''\rangle$ then from the first condition we get $J = J' = J''$, $L \supseteq L' \supseteq L''$ and $K \subseteq K' \subseteq K''$. Thus, the first condition holds for $\langle L, I, K, J\rangle$ and $\langle L'', I'', K'', J''\rangle$. Next, suppose $L \neq L''$. First suppose $L \neq L'$. In that case, $I \cap K' \neq \emptyset$ and since $K' \subseteq K''$ also $I \cap K'' \neq \emptyset$, and we are done. Otherwise, $L = L'$ and so it follows that $I \cup K = I' \cup K'$. If $K = K'$ then $\langle L, I, K, J\rangle = \langle L', I', K', J'\rangle$ and there is nothing to prove, so we can assume $K \neq K'$. But since the terminating positions are increasing we must have that the increase of K comes from the decrease of I so $I \cap K' \neq \emptyset$ and also $I \cap K'' \neq \emptyset$. □

In case the ranks of an operator can be linearly ordered, within the ranks there is no overlap of transition rules and, transition rules and termination rules are consistent it turns out that axiomatization is straightforward (as will be discussed below). Operators having these properties are referred to as smooth and distinctive.

Definition 16 A smooth n-ary operation f is called *smooth* and *distinctive* with respect to specification *TSS* if

- there is a ranking of the transition rules for f in *TSS*, such that the ordering $\succcurlyeq$ introduced in the previous definition becomes a total order;
- for any two distinct rules ρ, ρ' of the form (3) of the same rank there exists an active argument position i such that the corresponding action in the premise (a_i) is different;
- for each termination rule θ and each transition rule ρ for f in *TSS* it holds that some active argument position of ρ becomes terminating for θ.

For a smooth and distinctive operation f it holds that $neg(\rho) = neg(\rho')$ for any two transition rules ρ, ρ'. We define $neg(f) = neg(\rho)$ and $nonneg(f) = \{1, \ldots, n\} \setminus neg(\rho)$ where ρ is an arbitrary transition rule for f in *TS*. With respect to the total ordering of ranks, a rank of f is a rank of a transition rule of f.

Lemma 17 Suppose there is a ranking of the transition rules for a smooth operation f that makes it also distinctive. Then this ranking is unique.

Proof Consider the rule for f that is lowest in the ranking. Since the set of passive positions is decreasing, and every position is not passive somewhere, it must be that the set of passive positions in the low rule is empty. Thus, the ranking for this rule is fixed. Since the set of negative positions must be the same in every rule, this also fixes the ranking for all other rules. $\square$

As a consequence of this lemma, we can talk about the ranking of a smooth and distinctive operation f. If ρ is an action rule of f, we call its rank $\mathit{rank}(\rho)$. The intuition for the ordering on the transition rules for a smooth and distinctive n-ary operation f is the following: Suppose ρ and ρ' are two transition rules for f with $\mathit{rank}(\rho) \succcurlyeq \mathit{rank}(\rho')$. The ordering on $\succcurlyeq$ then demands that a passive position in ρ' must be passive in ρ as well and, conversely, that a terminating position in ρ must also be terminating in ρ'. Now, let $\rho_1 \succcurlyeq \ldots \succcurlyeq \rho_m$ be in descending order and $p \in \{1, \ldots, n\}$ a non-negative position in f, i.e. $p \in \mathit{nonneg}(f)$. The position p can either be passive, active or terminating in $\rho_1, \ldots, \rho_m$, but in view of the observation above we have that for suitable $0 \leq k \leq \ell \leq m$ it holds that $p \in \mathit{pass}(\rho_i)$ for $1 \leq i \leq k$, $p \in \mathit{act}(\rho_i)$ for $k < i \leq \ell$ and $p \in \mathit{term}(\rho_i)$ for $\ell < i \leq m$. So, in the context of $f(x_1, \ldots, x_n)$, the variable x_p at position p has a life-cycle typically from passive, via active, to terminating (but, possibly, p doesn't start out as passive, doesn't become active or doesn't reach the termination stage).

For a smooth and distinctive n-ary operation f we have that for closed terms of the form $f(t_1, \ldots, t_n)$ where each $t_i \equiv \varepsilon, \delta, a'.t'$ at most one of the transition rules for f applies: Suppose ρ and ρ' are two distinct rules that match $f(t_1, \ldots, t_n)$ (i.e. the terms t_i are such that the premises of both rules are satisfied). We either have $\mathit{rank}(\rho) = \mathit{rank}(\rho')$ or, without loss of generality, $\mathit{rank}(\rho) \succ \mathit{rank}(\rho')$ (meaning $\mathit{rank}(\rho) \succcurlyeq \mathit{rank}(\rho')$ and $\mathit{rank}(\rho) \neq \mathit{rank}(\rho')$. From the requirements of Definition 16 above we then obtain in the first case that for some $i \in \{1, \ldots, n\}$, $t_i \equiv a'.t'$ with $a' = a_i$ (the action of the i-th premise for ρ), $a' = a'_i$ (the action of the i-th premise for ρ') but also $a_i \neq a'_i$. For the second case we obtain from $\mathit{rank}(\rho) \succ \mathit{rank}(\rho')$ that $\mathit{act}(\rho) \cap \mathit{term}(\rho') \neq \emptyset$. So, for some $i \in \{1, \ldots, n\}$ we have $t_i \equiv a_i.t'$ as t_i matches the source of the i-th premise of ρ, but also $t_i \equiv \varepsilon$ as according to the rule ρ' the term t_i should terminate. All cases thus lead to a contradiction, and we conclude that $f(t_1, \ldots, t_n)$ does not match two distinct transition rules ρ and ρ'. By the last condition for distinctiveness in Definition 16 it is immediate that it cannot be the case for a smooth and distinctive operation f that both a transition rule ρ and a termination rule θ apply at the same time. However, overlapping of termination rules is allowed, as this does not effect the operational semantics nor the axiomatization.

If we consider only transition rules ρ with empty sets $\mathit{term}(\rho)$ and $\mathit{pass}(\rho)$, the notion of smooth and distinctive for the *tagh*-format specializes to this notion for *GSOS* as introduced in [ABV94]. Note that, in the absence of termination conditions, a non-active argument can be regarded as a negative one with an empty set of forbidden actions, so that the requirement for smoothness of an operation becomes trivial. In [ABV94] there is another requirement for smooth operations, viz. that the negative arguments of all transition rules coincide. In the set-up here, this is subsumed by the condition of total ordering for smooth and distinctive operations: if $\rho \not\succ \rho'$ we have $\mathit{neg}(\rho) = \mathit{neg}(\rho')$.

In the present setting there is for smooth rules the demand that the index set I is non-empty, which is not required by the definition of [ABV94]. The requirement of at least one active position in a smooth transition rule will be needed in our proof of the soundness of the distributive laws for negative arguments, introduced below and that are superfluous in the setting of [ABV94] but are essential for our treatment of termination (cf. Lemma 28). The condition is fulfilled in practically all cases. However, theoretically, an additional dummy argument can be added to an operation without active arguments in order to meet the non-emptiness condition of the index set I. The condition for a position p of a smooth operation to occur non-passively in some rule ρ will be needed in the proof of the head-normalization result Lemma 35. We stress that our primary aim is to deal with explicit termination as well as to allow for what we have baptized 'passive' variables, since this will lead, in many cases, to a more satisfactory axiomatization.

Examples 18

(a) The binary operation ';' of sequential composition comes equipped, in the set-up with explicit termination, with two transition rules and one termination rule:

$$(\mathit{Seq}_1)\ \frac{x \xrightarrow{a} x'}{x;y \xrightarrow{a} x';y} \qquad (\mathit{Seq}_2)\ \frac{x{\downarrow} \quad y \xrightarrow{a} y'}{x;y \xrightarrow{a} y'} \qquad (\mathit{Seq}_\varepsilon)\ \frac{x{\downarrow} \quad y{\downarrow}}{(x;y){\downarrow}}$$

We check that ';' in our set-up (in contrast to the one of [ABV94]) is a smooth and distinctive operation. The variable y in (Seq_1) must be considered a passive variable, since it is active in (Seq_2).

- We have $\mathit{rank}(\mathit{Seq}_1) = \langle \{2\}, \{1\}, \emptyset, \emptyset \rangle \succ \langle \emptyset, \{2\}, \{1\}, \emptyset \rangle = \mathit{rank}(\mathit{Seq}_2)$. So, this ranking makes the set of ranks totally ordered.
- There are no two distinct rules of equal rank. Hence the condition on actions is trivially satisfied.
- It holds that $\mathit{term}(\mathit{Seq}_\varepsilon) = \{1,2\}$ and $1 \in \mathit{act}(\mathit{Seq}_1)$, $2 \in \mathit{act}(\mathit{Seq}_2)$, so $\mathit{term}(\mathit{Seq}_\varepsilon) \cap \mathit{act}(\mathit{Seq}_i) \neq \emptyset$ for $i = 1, 2$.

(b) The binary operation '$\mathbin{|\!\lfloor}$', usually referred to as leftmerge, has one transition rule and one termination rule:

$$(\mathit{Leftmerge}_1)\ \frac{x \xrightarrow{a} x'}{x \mathbin{\|\!\!\lfloor} y \xrightarrow{a} x' \parallel y} \qquad\qquad (\mathit{Leftmerge}_\varepsilon)\ \frac{x{\downarrow} \quad y{\downarrow}}{(x \mathbin{\|\!\!\lfloor} y){\downarrow}}$$

Since there is only one transition rule, it cannot have passive variables, and y must be classified negative in $(\mathit{Leftmerge}_1)$. Therefore we have $\mathit{rank}(\mathit{Leftmerge}_1) = \langle \emptyset, \{1\}, \emptyset, \{2\} \rangle$. As the leftmerge has only one transition rule, it is clear that this ranking makes it a smooth and distinctive operation, since $\mathit{act}(\mathit{Leftmerge}_1) \cap \mathit{term}(\mathit{Leftmerge}_\varepsilon) = \{1\}$. (As an aside, the condition in the termination rule $(\mathit{Leftmerge}_\varepsilon)$ that the variable y is terminating is necessary to guarantee associativity of the related merge operation, cf. [Vra97].)

In concrete examples, such as the examples above, we prefer the usage of the more colloquial variable names like x, x', y, y', etc. instead of the technical x_1, y_1, x_2, y_2, etc., respectively. Also note that, in fact, we have transition schemes for (Seq_1), (Seq_2) and $(\mathit{Leftmerge}_1)$ rather than transition rules, as we have transition rules (Seq_1), (Seq_2) and $(\mathit{Leftmerge}_1)$, respectively, for each action $a \in \mathit{Act}$.

Before we are ready to describe the axioms generated for a smooth and distinctive n-ary operation f for a *tagh*-transition system, we need some notation and introduce the notion of a handle: Let $p \in \mathit{nonneg}(f)$. By smoothness of f there exists a transition rule ρ for f such that $p \notin \mathit{pass}(\rho)$. As the transition rules are linearly ordered by rank, there exists a transition rule ρ, maximal in rank but not necessarily unique, such that $p \notin \mathit{pass}(\rho)$. In that situation we put $\mathit{rank}(p) = \mathit{rank}(\rho)$ and $\mathit{act}(p) = \mathit{act}(\rho)$, $\mathit{neg}(p) = \mathit{neg}(\rho)$, etc. We have the following property:

Lemma 19 Let f be a smooth and distinctive n-ary operation and $p \in \{1, \ldots, n\}$ a non-negative position of f. If ρ is a transition rule for f with $p \in \mathit{pass}(\rho)$, then $\mathit{rank}(\rho) \succ \mathit{rank}(p)$.

Proof By definition, $\mathit{rank}(p) = \max\{ \mathit{rank}(\rho') \mid p \notin \mathit{pass}(\rho') \}$. Note that this is well-defined: by smoothness of f the set at the right-hand side is non-empty; by distinctiveness its maximum exists. Let ρ' be such that $\mathit{rank}(p) = \mathit{rank}(\rho')$. Clearly $\mathit{rank}(\rho) \neq \mathit{rank}(\rho')$ as $p \in \mathit{pass}(\rho), p \notin \mathit{pass}(\rho')$. Also $\mathit{rank}(\rho) \not\prec \mathit{rank}(\rho')$ for the same reason. Hence, by the totality of the ordering $\succcurlyeq$ it follows that $\mathit{rank}(\rho) \succ \mathit{rank}(\rho')$, thus $\mathit{rank}(\rho) \succ \mathit{rank}(p)$. □

Also, if, for a 4-tuple of index sets R, we have that $R = \mathit{rank}(\rho)$, we put $\mathit{act}(R) = \mathit{act}(\rho)$, $\mathit{neg}(R) = \mathit{neg}(\rho)$, etc. The index set $\mathit{handle}(p)$, the handle of p with respect to f and *TS*, is defined as $\mathit{term}(p)$ if $p \in \mathit{nonneg}(f)$, and as $\mathit{nonneg}(f)$ if $p \in \mathit{neg}(f)$.

The idea behind the notion of a handle is that for a smooth operation f and non-negative position $p \in \{1, \ldots, n\}$ the set $\mathit{handle}(p)$ consists of all positions that are required to be terminating when the position p becomes non-passive, i.e.,

$$\mathit{handle}(p) = \bigcap \{\, \mathit{term}(\rho) \mid p \notin \mathit{pass}(\rho), \rho \text{ transition rule for } f \,\}$$

For a negative position p for f, $\mathit{handle}(p)$ simply consists of all non-negative positions. The handles are used in the formulation of distributivity laws; the subset-ordering on the handles of an operation induces an ordering on the applicability of these laws.

The next definition describes the various laws associated with a smooth and distinctive operation.

Definition 20 Let f be a smooth and distinctive n-ary operation for a *tagh*-transition system specification *TSS*.

(a) For a position $p \in \{1, \ldots, n\}$ the distributive law for p with respect to f is given as follows:

$$f(\zeta_1, \ldots, z'_p + z''_p, \ldots, \zeta_n) = f(\zeta_1, \ldots, z'_p, \ldots, \zeta_n) + f(\zeta_1, \ldots, z''_p, \ldots, \zeta_n) \tag{4}$$

where $\zeta_q \equiv \varepsilon$ for $q \in \mathit{handle}(p)$ and $\zeta_q \equiv z_q$ for $q \notin \mathit{handle}(p), q \neq p$. The distributive laws of f are the laws (4) for all positions p of f.

(b) For a transition rule ρ of the format (3) the action law for ρ is given as follows:

$$f(\zeta_1, \ldots, \zeta_n) = a.C[z'_i, z_j, z_\ell\,] \tag{5}$$

where $\zeta_i \equiv a_i.z'_i$ for $i \in \mathit{act}(\rho)$, $\zeta_j \equiv \partial^1_{B_j}(z_j)$ for $j \in \mathit{neg}(\rho)$ with $B_j \neq \emptyset$ and $\zeta_j \equiv z_j$ for $j \in \mathit{neg}(\rho)$ with $B_j = \emptyset$, $\zeta_k \equiv \varepsilon$ for $k \in \mathit{term}(\rho)$ and $\zeta_\ell \equiv z_\ell$ for $\ell \in \mathit{pass}(\rho)$. The action laws of f are all the laws (5) for all transition rules ρ for f.

(c) For a rank R for f the deadlock laws are given as follows:

$$f(\zeta_1, \ldots, \zeta_n) = \delta \tag{6}$$

where ζ_m is of the form ε, δ or $a'_m.z'_m$ for $m \in \mathit{act}(R) \cup \mathit{term}(R)$, ζ_j is of the form z_j, δ, $b'_j.z'_j$ or $z_j + b'_j.z_j$ for $j \in \mathit{neg}(R)$ and $\zeta_\ell \equiv z_\ell$ for $\ell \in \mathit{pass}(R)$ such that, for each rule ρ for f in *TS* of the format (3), there exists a position p such that one of the following cases holds:

* $p \in \mathit{act}(\rho)$ and $\zeta_p \equiv \varepsilon$, $\zeta_p \equiv \delta$ or $\zeta_p \equiv a'_p.z'_p$ with $a'_p \neq a_p$, or
* $p \in \mathit{neg}(\rho)$ and $\zeta_p \equiv b'_p.z'_p$ or $\zeta_p \equiv z_p + b'_p.z'_p$ with $b'_p \in B_p$, or
* $p \in \mathit{term}(\rho)$ and $\zeta_p \equiv \delta$ or $\zeta_p \equiv a'_p.z'_p$,

and, for each termination rule θ for f there exists a position $p \in \{1, \ldots, n\}$ such that $\zeta_p \equiv \delta$ or $\zeta_p \equiv a'_p.z'_p$. The deadlock laws of f are all the laws (6) for all ranks R for f.

(d) For a termination rule θ for f the termination law for θ is given as follows:

$$f(\zeta_1, \ldots, \zeta_n) = \varepsilon \tag{7}$$

where $\zeta_p \equiv \varepsilon$ for $p \in \mathit{term}(\theta)$ and $\zeta_p \equiv z_p$ for $p \notin \mathit{term}(\theta)$. The termination laws of f are all the laws (7) for all termination rules θ for f.

In a distributive law of an operation f with position p we demand a 'fingerprint of ε-s' for p instead of allowing a variable for *handle*-arguments. This way, non-determinism at a position is only resolved if it is guaranteed that there is sufficient termination at other positions, as will be illustrated in the examples for sequential composition ';' and leftmerge '$\lfloor\!\lfloor$' below. Note that there is also a distributive law for negative positions (which is not present in the approach of [ABV94]). The action laws are similar to those of [ABV94]. Here, we also adopt the difference in the handling of a non-empty or empty set of negative actions B_j. For the deadlock laws, it should be syntactically guaranteed that no transition rule will match. If such can be established without instantiating passive arguments, this can be reflected by the rule having variables at those places. It should however be ascertained by the form of the term that no termination rule will apply as well. The termination laws themselves are straightforward translations of the corresponding termination rules.

Examples 21

(a) The transition and termination rules for ';', as given in Example 18, generate, according to the definitions above, the following equations:

$$\begin{array}{rlcrl} (x_1 + x_2); y &= (x_1; y) + (x_2; y) & \qquad & \varepsilon; \delta &= \delta \\ \varepsilon; (y_1 + y_2) &= (\varepsilon; y_1) + (\varepsilon; y_2) & & \delta; y &= \delta \\ (a.x'); y &= a.(x'; y) & & \varepsilon; \varepsilon &= \varepsilon \\ \varepsilon; (a.y') &= a.y' & & & \end{array}$$

Note that, apart from the equation $\delta; y = \delta$, the operation ';' has also other deadlock laws, viz. $\delta; \varepsilon = \delta$, $\delta; (a.y') = \delta$ and $\delta; (y + b.y')$, which are special cases of the displayed law $\delta; y = \delta$.

(b) Similarly, we obtain for the leftmerge '$\lfloor\!\lfloor$' as characterized by the rules in

Example 18b the following axiom system:

$$
\begin{array}{rclr}
(x_1 + x_2) \mathbin{⌊\!⌊} y & = & (x_1 \mathbin{⌊\!⌊} y) + (x_2 \mathbin{⌊\!⌊} y) & \varepsilon \mathbin{⌊\!⌊} \delta = \delta \\
\varepsilon \mathbin{⌊\!⌊} (y_1 + y_2) & = & (\varepsilon \mathbin{⌊\!⌊} y_1) + (\varepsilon \mathbin{⌊\!⌊} y_2) & \varepsilon \mathbin{⌊\!⌊} (b.y') = \delta \\
(a.x') \mathbin{⌊\!⌊} y & = & a.(x' \parallel y) & \delta \mathbin{⌊\!⌊} y = \delta \\
 & & & \varepsilon \mathbin{⌊\!⌊} \varepsilon = \varepsilon
\end{array}
$$

Again we omit the superfluous instantiations of the axiom $\delta \mathbin{⌊\!⌊} y = \delta$. Note that actually we have exactly the existing axiomatization of [Vra97].

From the termination laws $\varepsilon; \varepsilon = \varepsilon$ and $\varepsilon \mathbin{⌊\!⌊} \varepsilon = \varepsilon$ in the examples above, one can see the necessity of a distributive law for a negative argument, here in both cases the second position. Without these distributive laws it is not possible to derive, e.g., $\varepsilon; (a.t + \varepsilon) = a.t + \varepsilon$ and $\varepsilon \mathbin{⌊\!⌊} (a.t + \varepsilon) = \varepsilon$, which is desired for our interpretation of optional termination. Another observation here is that the handles indicate which distributivity law should be applied first in a rewriting procedure. In the case of the sequential composition ';' given by the rules in Example 18 we have that *handle*$(1) = \emptyset$, *handle*$(2) = \{1\}$. The distributivity law for the second position is only applicable when the term at the first position is terminating and hence not a sum, i.e. not a choice between two subprocesses.

The disrupt or disabling operator '$\gg$' is well-known, e.g., from LOTOS [Bri89] (see also [BB00]). In the process $x \gg y$ the subprocess x may proceed, unless the subprocess y takes over control. It terminates when either of the subprocesses does so. Thus, the disrupt operator has the following operational rules:

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

The disrupt operator can be given a ranking which makes it a smooth operation. However, it cannot be made distinctive as well. For, if it is distinctive, then it has a ranking which is a total order. But, if the left transition rule has higher rank than the right transition rule, then the active x in the left rule must be terminated in the right rule, which is not the case. Likewise the right rule cannot be of higher rank than the left rule. The two transition rules cannot be of equal rank as the active variables differ. Hence the rules cannot be ranked as a total order and the operator '$\gg$' is not distinctive.

However, if we split the operation '$\gg$' into two, introducing '$\gg_1$' and '$\gg_2$' say, then there is a ranking which makes both operations smooth and distinctive:

$$\frac{x \xrightarrow{a} x'}{x \gg_1 y \xrightarrow{a} x' \gg y} \qquad \frac{x{\downarrow}}{(x \gg_1 y){\downarrow}} \qquad \frac{y \xrightarrow{a} y'}{x \gg_2 y \xrightarrow{a} y'} \qquad \frac{y{\downarrow}}{(x \gg_2 y){\downarrow}}$$

The idea of splitting up '$\gg$' is also present in the transition system specification for this operation in [BB00]. The relationship between the various disrupt operations is expressed by the law $x \gg y = (x \gg_1 y) + (x \gg_2 y)$. Another instance of this trick is the well-known representation of the merge '$\|$' in terms of leftmerge '$\lfloor\!\lfloor$', rightmerge '$\rfloor\!\rfloor$' and communication merge '$\mid$' using the law $x \| y = (x \lfloor\!\lfloor y) + (x \rfloor\!\rfloor y) + (x \mid y)$ (cf. [BK84]).

The same approach, as pointed out in [ABV94] and also applicable for the *tagh*-format, of partitioning the set of transition rules and introducing smooth and distinctive suboperations works in general to split a smooth but non-distinctive operation f into a number of smooth and distinctive ones, $f_1, \ldots, f_s$ say. Here we only present how the resulting equations can be derived. The soundness of this law is proven below, see Lemma 32.

Definition 22 Let f be a smooth but non-distinctive n-ary operation for the *tagh*-transition system specification *TSS*. The n-ary operations $f_1, \ldots, f_s$ (all distinct) are called distinctive versions of f in a disjoint extension TSS' of *TSS* if the transition rules and termination rules for each f_r in TSS' $(1 \leq r \leq s)$ form, after renaming of f_r in the source of the rules by f, a partitioning of all the rules for f in *TSS*. The equation

$$f(\vec{z}) = f_1(\vec{z}) + \cdots + f_s(\vec{z}) \tag{8}$$

is then referred to as the distinctivity law for f.

Note that the extended transition system specification TSS' contains all rules for f as present in *TSS*.

The previous definition addresses smooth but non-distinctive operations. However, some operations are not smooth at all. There may be several ways in which the transition rules of an operation f can violate the various conditions of the definition of smooth operations: there can be a transition rule for f that is not of the format (3), thus, either

(i) there are multiple premises for an action-argument, or
(ii) there is overlap of the index sets, or
(iii) the target contains 'trespassing' variables, i.e. in C occurs a variable x_p with $p \in I$ or $p \in K$, or
(iv) there is no active premise, or
(v) the given ranking has a position p for which there is no transition rule for f for which this p is non-passive.

In the last case, we can make a passive position into a negative one: If a position p occurs passively only in the transition rules of an operation f we may be able interpret p as a *negative* position with an empty set of forbidden transitions. Thus removing p from the index set L and adding it to the index set J.

Example 23 Consider the operator f given in Example 13. This is a smooth but non-distinctive operator as its ranks (no matter how the rankings for the four rules are chosen) do not form a total order. As rules (1) and (3) have each four possible rankings and rules (2) and (4) have each two, there are in total 64 different ways to classify the transition rules.

Suppose the rules (1) to (4) are ranked as $\langle\{y,z\},\{x\},\emptyset,\emptyset\rangle$, $\langle\{z\},\{y\},\{x\},\emptyset\rangle$, $\langle\{y,z\},\{x\},\emptyset,\emptyset\rangle$ and $\langle\{y\},\{z\},\{x\},\emptyset\rangle$, respectively. Then we can take, e.g., rules (1) to (3) together and set rule (4) apart. Then we introduce two auxiliary operators with the following rules:

$$\frac{x \xrightarrow{a} x'}{g(x,y,z) \xrightarrow{a} A}\ (1') \qquad \frac{x\!\downarrow\ y \xrightarrow{b} y'}{g(x,y,z) \xrightarrow{b} B}\ (2') \qquad \frac{x \xrightarrow{c} x'}{g(x,y,z) \xrightarrow{c} C}\ (3')$$

$$\frac{x\!\downarrow\ z \xrightarrow{d} z'}{h(x,y,z) \xrightarrow{d} D}\ (4')$$

As the variable z does not occur in any premise, the operator g would not be smooth if we interpret z as passive. Therefore, we classify z as negative. Then rules $(1')$ and $(3')$ have rank $\langle\{y\},\{x\},\emptyset,\{z\}\rangle$ and rules $(2')$ has rank $\langle\emptyset,\{y\},\{x\},\{z\}\rangle$. As $\langle\{y\},\{x\},\emptyset,\{z\} \preccurlyeq \langle\emptyset,\{y\},\{x\},\{z\}\rangle$ the ranks are totally ordered and it follow that g is a smooth and distinctive operator (with respect to the chosen ranking). Similar to the case of g, the variable y of the operator h should be regarded as negative for h to be smooth. Its ranking trivially constitutes a total order. Note that there is no other option in making h smooth (and distinctive) than by the negative interpretation of y. This is a typical situation. If we choose, for example, to split up f into two operators g' and h' given by

$$\frac{x \xrightarrow{a} x'}{g'(x,y,z) \xrightarrow{a} A}\ (1'') \qquad \frac{x \xrightarrow{c} x'}{h'(x,y,z) \xrightarrow{c} C}\ (3'')$$

$$\frac{x\!\downarrow\ y \xrightarrow{b} y'}{g'(x,y,z) \xrightarrow{b} B}\ (2'') \qquad \frac{x\!\downarrow\ z \xrightarrow{d} z'}{h'(x,y,z) \xrightarrow{d} D}\ (4'')$$

the same phenomena of negative interpretation occur.

If a transition rule for an n-ary operation f has an empty set of active premises (case (iv) above), we can consider an $n+1$-ary operation f' obtained from f by adding a dummy variable x_0. For the dummy variable we require a dummy transition. By extending the transition system specification with a constant Ω, say, with transition rule

$$\frac{\emptyset}{\Omega \xrightarrow{\omega} \Omega}$$

instantiation of the dummy variable with Ω in $f'(x_0, x_1, \ldots, x_n)$ will yield a term bisimilar to $f(x_1, \ldots, x_n)$. We therefore add the law $f(x_1, \ldots, x_n) = f'(\Omega, x_1, \ldots, x_n)$ to the equational theory. Note that addition to the transition system specification of the smooth and distinctive constant Ω with the transition rule as its only rule will yield the axiom $\Omega = \omega.\Omega$ in the corresponding equational theory.

Let us consider, in order to illustrate this, the so-called don't care choice denoted by '$\oplus$'. It is modeled by the transition rules with no premises below. Therefore we interpret the first and second position to occur negatively in the two rules.

$$\frac{\emptyset}{x \oplus y \xrightarrow{\ell} x} \qquad\qquad \frac{\emptyset}{x \oplus y \xrightarrow{r} y}$$

This way the operation '$\oplus$' is not smooth. It is lacking an active premise. The defect, though, can be overcome easily; we simply add a dummy variable and extend the transition system with a fresh constant Ω with only an $\Omega \xrightarrow{\omega} \Omega$-transition and expand the equational theory with the ω-law $x \oplus y = \oplus'(\Omega, x, y)$. This will not contribute essentially to the dynamics of the operation '$\oplus'$' compared to '$\oplus$', nor to its termination behavior. We thus arrive at

$$\frac{w \xrightarrow{\omega} w'}{\oplus'(w, x, y) \xrightarrow{\ell} x} \qquad\qquad \frac{w \xrightarrow{\omega} w'}{\oplus'(w, x, y) \xrightarrow{r} y}$$

Now, both the first and second position of '$\oplus$' are negative and the adapted left and right rule both have an active transition. Thus '$\oplus$' is a smoothened version of the operation '$\oplus$'.

To illustrate the countermeasure for multiple active transitions, overlap over index sets and trespassing variable in the target, consider the following, synthesized, one-rule transition system adapted from [ABV94]. The operation f is non-smooth because there are multiple transitions for an active variable (viz. $x \xrightarrow{a} y_1$ and $x \xrightarrow{b} y_2$), the active and terminating variable x occurs in the target $x + y_1$, the index sets overlap (its only position 1 occurs both as active, as terminating as well as negative argument).

$$\frac{x \xrightarrow{a} y_1 \quad x \xrightarrow{b} y_2 \quad x \overset{c}{\not\rightarrow} \quad x{\downarrow}}{f(x) \xrightarrow{d} x + y_1} \qquad\qquad \frac{x{\downarrow}}{f(x){\downarrow}}$$

The key idea is not to split f into new operations, but to split the variable x into new variables, i.e., we introduce separate copies x_1, x_2, x_3, x_4 of the variable x to relieve the overlap and multiplicity. The rules for f are translated into rules for a fresh operation f'. This yields the following transition system for which f' is a smooth operation:

$$\frac{x_1 \xrightarrow{a} y_1 \quad x_2 \xrightarrow{b} y_2 \quad x_3 \overset{c}{\not\rightarrow} \quad x_4{\downarrow}}{f'(x_1, x_2, x_3, x_4) \xrightarrow{d} x_3 + y_1} \qquad\qquad \frac{x_1{\downarrow} \quad x_2{\downarrow} \quad x_3{\downarrow} \quad x_4{\downarrow}}{f'(x_1, x_2, x_3, x_4){\downarrow}}$$

As connecting law for f we have $f(x) = f'(x, x, x, x)$ which enforces that in the right-hand side we indeed have copies of the original argument.

In the next definition we will formalize the idea in a more general case. In the presentation below we introduce mappings ϕ and ψ to make the correspondence explicit between a variable x_i and its splittings $\{\, x'_{i'} \mid \phi(i') = i \,\}$ and the actions a_{ip} and output variables y_{ip} and their new names $a'_{i'}$ and $y'_{i'}$ with $\psi(i') = (i, p)$.

Definition 24 Let f be a non-smooth n-ary operation of a *tagh*-transition system specification *TSS*. The m-ary operation f' is called the smooth version of f in a disjoint extension *TSS'* of *TSS*, if mappings $\phi\colon \{\, 1, \ldots, m\} \to \{1, \ldots, n\}$ and $\psi\colon \{1, \ldots, m\} \to \{1, \ldots, n\} \times \{1, \ldots, m\}$ and a 1–1 correspondence between the rules of f and f' exist, such that

(a) a transition rule ρ for f in *TSS* of the form

$$\frac{\{\, x_i \xrightarrow{a_{ip}} y_{ip} \mid i \in I, p \in P_i \,\} \quad \{\, x_j \overset{b_{jq}}{\not\rightarrow} \mid j \in J, q \in Q_j \,\} \quad \{\, x_k{\downarrow} \mid k \in K \,\}}{f(x_1, \ldots, x_n) \xrightarrow{a} C[x_i, x_j, x_k, x_\ell, y_{ip}]} \tag{9}$$

corresponds to a smooth transition rule ρ' for f' in *TSS'* of the form

$$\frac{\{\, x'_i \xrightarrow{a'_i} y'_i \mid i \in I' \,\} \quad \{\, x'_j \overset{b'_{jq}}{\not\rightarrow} \mid j \in J', q \in Q'_j \,\} \quad \{\, x'_k{\downarrow} \mid k \in K' \,\}}{f'(x'_1, \ldots, x'_m) \xrightarrow{a} C'[x'_j, x'_\ell, y'_{ip}]} \tag{10}$$

such that the mapping $x'_i \xrightarrow{a'_i} y'_i \mapsto x_{\phi(i)} \xrightarrow{a'_i} y_{\psi(i)}$, $x'_j \overset{b'_{jq}}{\not\rightarrow} \mapsto x_{\phi(j)} \overset{b'_{jq}}{\not\rightarrow}$, $x'_k{\downarrow} \mapsto$

$x_{\phi(k)}\!\downarrow$ is a bijection between the premises of ρ and the premises of ρ' and $C[x_i, x_j, x_k, x_\ell, y_{ip}] \equiv \chi(C'[x'_j, x'_\ell, y'_i])$ for a substitution χ with $\chi(x'_j) = x_{\phi(j)}, \chi(x'_\ell) = x_{\phi(\ell)}, \chi(y'_i) = y_{\psi(i)}$,

(b) a termination rule θ for f in *TSS* of the form on the left below corresponds to a termination rule for f' in *TSS*$'$ of the form on the right below

$$\frac{\{\, x_k\!\downarrow \mid k \in K \,\}}{f(x_1, \ldots, x_n)\!\downarrow} \qquad\qquad \frac{\{\, x'_k\!\downarrow \mid k \in K' \,\}}{f'(x'_1, \ldots, x'_m)\!\downarrow}$$

where $K' = \phi^{-1}(K)$.

The equation

$$f(z_1, \ldots, z_n) = f'(\zeta_1, \ldots, \zeta_m), \tag{11}$$

with $\zeta_p \equiv z_{\phi(p)}$ for $p \in \{1, \ldots, n\}$, is called the smoothening law for f. In case the index set I' is empty, f' will be an $m+1$-ary operation and to its transition rules we add the active premise $x'_0 \xrightarrow{\omega} y'_0$. The transition system specification *TSS*$'$ is assumed to contain the transition $\Omega \xrightarrow{\omega} \Omega$ as the only transition for the label ω. In this case the equation

$$f(z_1, \ldots, z_n) = f'(\Omega, \zeta_1, \ldots, \zeta_m), \tag{12}$$

is referred to as the smoothening law for f.

Example 25 The 'standard' example of a non-smooth operation is the priority operator θ of [BBK86]. Assuming a partial ordering on '$>$' on Act, the action rules of the unary θ and its binary smoothening θ' are the following:

$$\frac{x \xrightarrow{a} x' \quad \{\, x \stackrel{b}{\nrightarrow} \mid b > a \,\}}{\theta(x) \xrightarrow{a} \theta(x')} \qquad \frac{x\!\downarrow}{\theta(x)\!\downarrow}$$

$$\frac{x \xrightarrow{a} x' \quad y \stackrel{b}{\nrightarrow} \quad (b > a)}{\theta'(x, y) \xrightarrow{a} \theta(x')} \qquad \frac{x\!\downarrow \quad y\!\downarrow}{\theta'(x, y)\!\downarrow}$$

The smoothening law for the priority operator θ is $\theta(x) = \theta'(x, x)$.

In the above we have defined how to transform a non-smooth operation into a smooth one and how to split a smooth but non-distinctive operation into several smooth and distinctive ones. In these situations the transition system specification will be extended disjointly, i.e., the dynamics and termination of operations already in the specification remain unaffected. Also we have defined the smoothening law (11) and its variant (12) and the distinctivity law (8) that connects the original and new operations. For smooth and distinctive operations we have introduced various equations describing distributivity, dynamics, deadlock and termination. This all together induces the notion of the transition system specification and the set of equations generated by a *tagh*-transition system specification.

Definition 26 Let *TSS* be a *tagh*-transition system specification. The *tagh*-transition system specification TSS' generated by *TSS* and the equational theory ET' generated by *TSS* are given by the following procedure:

Step 1 Let TSS_1 be the disjoint extension of *TSS* with TSS_∂. Let ET_1 contain the equations for '+' and '∂_B^1' as given in Section 2.

Step 2 For every non-smooth operation f of *TSS* extend TSS_1 with the smooth version f' of f and add to ET_2 the corresponding smoothening law (11) or (12). This yields TSS_2 and ET_2.

Step 3 For every smooth but non-distinctive operation f of TSS_2 different from '+', extend TSS_2 with the distinctive versions $f_1, \ldots, f_s$ and add to ET_2 the distinctivity law (8). This yields TSS' and ET_3.

Step 4 For each smooth and distinctive operation f of TSS' not in TSS_∂, add to ET_3 the distributive laws (4), the action laws (5), the deadlock laws (6) and the termination laws (7). This yields ET'.

Examples 27 Application of the above procedure yields for the disrupt operator '$\gg$' and the priority operator θ the following generated equational theories:

$$
\begin{array}{rlrl}
x \gg y &= (x \gg_1 y) + (x \gg_2 y) & \delta \gg_1 y &= \delta \\
(x_1 + x_2) \gg_1 y &= (x_1 \gg_1 y) + (x_2 \gg_1 y) & \varepsilon \gg_1 y &= \varepsilon \\
(a.x') \gg_1 y &= a.(x' \gg y) & & \text{similar rules for '}\gg_2\text{'}
\end{array}
$$

$$
\begin{array}{rlrl}
\theta(x) &= \theta'(x, x) & \theta'(\delta, y) &= \delta \\
\theta'(x_1 + x_2, y) &= \theta'(x_1, y) + \theta'(x_2, y) & \theta'(\varepsilon, b.y') &= \delta \\
\theta'(\varepsilon, y_1 + y_2) &= \theta'(\varepsilon, y_1) + \theta'(\varepsilon, y_2) & \theta'(x, \delta) &= \delta \\
\theta'(a.x', \partial^1_{\{b \mid b > a\}}(y)) &= a.\theta(x') & \theta'(a.x, \varepsilon) &= \delta \\
\theta'(a.x', b.y + z) &= \delta \ \text{ if } b > a & \theta'(\varepsilon, \varepsilon) &= \varepsilon
\end{array}
$$

Note that the above axiomatizations are quite natural and improve upon the corresponding theory synthesized in [ABV94]. The equations for the disrupt operation coincide with those of [BB00]. The axiomatization for the priority operator avoids equations for the auxiliary 'unless' operation '$\triangleleft$' (cf. [BBK86]). However, one may want, as also discussed in [ABV94], to optimize the equations regarding their rewriting properties by introducing a rule $x \gg_2 y = y$ or to replace $\theta'(a.x', \partial^1_{\{b \mid b > a\}}(y)) = a.\theta(x')$ by the laws $\theta'(a.x, b.y + z) = \theta'(a.x, z)$ if $b \not> a$, $\theta'(a.x, \varepsilon) = a.\theta(x)$ and $\theta'(a.x, \delta) = a.\theta(x)$.

Coming back to the operator f of Example 13 again, recall that there are many ways of splitting up the transition rules of f in groups of one to three rules. Calculations indicate that all splittings lead to equivalent axiomatizations. We conjecture that this holds in general: the theory of equality of the original transition system specification is independent of the chosen splitting of smooth but non-distinctive operators.

Coming back to the operator f of Example 13 again, recall that there are many ways of splitting up the transition rules of f in groups of one to three rules, leading to axiomatizations with different auxiliary operations. From the results of the sequel, it follows that this leads to the same equational theory for closed terms: two closed terms over the original signature (without auxiliary operations) can be proved equal by the one extended equational theory exactly when this is the case for the other extended equational theory. We conjecture that this is also the case for equations of open terms, but leave a further investigation and proof of this as future work.

4 Soundness

In this section we first address the soundness of the laws generated for a smooth and distinctive operation: distributive laws, action laws, deadlock laws and termination laws. Next, we address the distinctivity law for a smooth but non-distinctive operation and the smoothening law for a non-smooth operation. Taking all results together we obtain a soundness result for the generated equational theory with respect to the transition system space induced by the generated disjoint extension of the original specification.

As a direct consequence of the incorporation of explicit termination in our set-up, both in the form of termination rules and in the form of having the possibility for termination premises in a transition rule, the proofs presented in this and in the next section are, at places, technically more involved as compared to [ABV94]. In particular, there are more cases in the analysis of arguments, and our format requires distributive laws for negative positions and also for termination laws (both are not present in the framework of Aceto et al.). The latter is necessary to deal with termination, as was illustrated by the leftmerge law $\varepsilon \mathbin{\|\!\_} \varepsilon = \varepsilon$ above.

Lemma 28 Let f be a smooth and distinctive n-ary operation of a *tagh*-transition system specification *TSS*. Then it holds, that the distributive laws (4) for every $p \in \{ 1, \ldots, n \}$ with respect to f are sound.

Proof Without loss of generality we can assume that f is a non-constant operation.

(a) Suppose $p \in \mathit{nonneg}(f)$ and it holds that $f(t_1, \ldots, t'_p + t''_p, \ldots, t_n) = f(t_1, \ldots, t'_p, \ldots, t_n) + f(t_1, \ldots, t''_p, \ldots, t_n)$ is a closed instance of the distributive law. So $t_q \equiv \varepsilon$ for $q \in \mathit{handle}(p)$. Let '$\succcurlyeq$' denote a total ordering relation on the ranks of f. We use a first step analysis and Lemma 3 to show that the terms $f(t_1, \ldots, t'_p + t''_p, \ldots, t_n)$ and $f(t_1, \ldots, t'_p, \ldots, t_n) + f(t_1, \ldots, t''_p, \ldots, t_n)$ are bisimilar in the induced transition system space.

(i) Assume the left-hand side admits a transition $f(t_1, \ldots, t'_p + t''_p, \ldots, t_n) \xrightarrow{a} t$ via the transition rule ρ of f for some action a and closed term t. It holds that $p \notin \mathit{pass}(\rho)$: For, if $p \in \mathit{pass}(\rho)$, then $\mathit{rank}(\rho) \succ \mathit{rank}(p)$. So, as by definition $\mathit{handle}(p) = \mathit{term}(p)$, we can choose $q \in \mathit{act}(\rho) \cap \mathit{term}(p)$. Then, on the one hand, $t_q \xrightarrow{a_p} t'_q$ for suitable t'_q, but, on the other hand, $q \in \mathit{handle}(p)$ and $t_q \equiv \varepsilon$. This is a contradiction. So $p \notin \mathit{pass}(\rho)$ and thus either $p \in \mathit{act}(\rho)$ or $p \in \mathit{term}(\rho)$.

Suppose $p \in \mathit{act}(\rho)$. Then $t'_p + t''_p \xrightarrow{a_m} t_p$ for some t_p. By inspection of the transition rules for '+' we derive that $t'_p \xrightarrow{a_p} t_p$ or $t''_p \xrightarrow{a_p} t_p$. Since all other premises of ρ with respect to $f(t_1, \ldots, t'_p + t''_p, \ldots, t_n)$, $f(t_1, \ldots, t'_p, \ldots, t_n)$ and $f(t_1, \ldots, t''_p, \ldots, t_n)$ are the same, it follows that $f(t_1, \ldots, t'_p, \ldots, t_n) \xrightarrow{a} t$ or $f(t_1, \ldots, t''_p, \ldots, t_n) \xrightarrow{a} t$ and thus $f(t_1, \ldots, t'_p, \ldots, t_n) + f(t_1, \ldots, t''_p, \ldots, t_n) \xrightarrow{a} t$.

Suppose $p \in \mathit{term}(\rho)$. Then we have that $(t'_p + t''_p){\downarrow}$. It follows by definition of '$\downarrow$' for '+' that $t'_p{\downarrow}$ or $t''_p{\downarrow}$. Since all other premises of ρ with respect to $f(t_1, \ldots, t'_p + t''_p, \ldots, t_n)$, $f(t_1, \ldots, t'_p, \ldots, t_n)$ and $f(t_1, \ldots, t''_p, \ldots, t_n)$ are the same, we derive that $f(t_1, \ldots, t'_p, \ldots, t_n) \xrightarrow{a} t$ or $f(t_1, \ldots, t''_p, \ldots, t_n) \xrightarrow{a} t$ and thus $f(t_1, \ldots, t'_p, \ldots, t_n) + f(t_1, \ldots, t''_p, \ldots, t_n) \xrightarrow{a} t$.

(ii) Assume that there is a transition $f(t_1, \ldots, t'_p, \ldots, t_n) + f(t_1, \ldots, t''_p, \ldots, t_n) \xrightarrow{a} t$ via the transition rule ρ for some a and t. By inspection of the transition system specification TSS_∂ as given in Section 2, it follows that $f(t_1, \ldots, t'_p, \ldots, t_n) \xrightarrow{a} t$ or $f(t_1, \ldots, t''_p, \ldots, t_n) \xrightarrow{a} t$. Without loss of generality we can assume $f(t_1, \ldots, t'_p, \ldots, t_n) \xrightarrow{a} t$. As before it holds that $p \in \mathit{act}(\rho)$ or $p \in \mathit{term}(\rho)$.

Suppose $p \in \mathit{act}(\rho)$. Then $t'_p \xrightarrow{a_p} t_p$ for some t_p. So $t'_p + t''_p \xrightarrow{a_p} t_p$. As the premises for positions different from p coincide, we obtain $f(t_1, \ldots, t'_p + t''_p, \ldots, t_n) \xrightarrow{a} t$. Suppose $p \in \mathit{term}(\rho)$. Then $t'_p{\downarrow}$. So, by definition of '$\downarrow$' for '+', $(t'_p + t''_p){\downarrow}$, hence, as all other premises for ρ with respect to $f(t_1, \ldots, t'_p + t''_p, \ldots, t_n)$ are satisfied, it follows that $f(t_1, \ldots, t'_p + t''_p, \ldots, t_n) \xrightarrow{a} t$.

(iii) Suppose $f(t_1, \ldots, t'_p + t''_p, \ldots, t_n){\downarrow}$ by some termination rule θ. If $p \notin \mathit{term}(\theta)$, then also $f(t_1, \ldots, t'_p, \ldots, t_n){\downarrow}$ and $f(t_1, \ldots, t''_p, \ldots, t_n){\downarrow}$, so $f(t_1, \ldots, t'_p, \ldots, t_n) + f(t_1, \ldots, t''_p, \ldots, t_n){\downarrow}$. Suppose $p \in \mathit{term}(\theta)$. Then we have that $(t'_p + t''_p){\downarrow}$ and $t_q{\downarrow}$ for $q \in \mathit{term}(\theta) \setminus \{p\}$. It follows by definition of $\downarrow$ for '+' that $t'_p{\downarrow}$ or $t''_p{\downarrow}$. So, by application of θ, either $f(t_1, \ldots, t'_p, \ldots, t_n){\downarrow}$ or $f(t_1, \ldots, t''_p, \ldots, t_n){\downarrow}$. Again by definition of $\downarrow$ for '+', we obtain $f(t_1, \ldots, t'_p, \ldots, t_n) + f(t_1, \ldots, t''_p, \ldots, t_n){\downarrow}$.

Suppose $f(t_1, \ldots, t'_p, \ldots, t_n) + f(t_1, \ldots, t''_p, \ldots, t_n){\downarrow}$. By definition of $\downarrow$

for $+$, we then have $f(t_1,\ldots,t'_p,\ldots,t_n){\downarrow}$ or $f(t_1,\ldots,t''_p,\ldots,t_n){\downarrow}$. Assume $f(t_1,\ldots,t'_p,\ldots,t_n){\downarrow}$ by application of the termination rule θ. If $p \notin \mathit{term}(\theta)$ then also $f(t_1,\ldots,t'_p+t''_p,\ldots,t_n){\downarrow}$ by application of θ. If $p \in \mathit{term}(\theta)$, we have $t'_q{\downarrow}$ and $t_q{\downarrow}$ for $q \in \mathit{term}(\theta)\setminus\{p\}$. We conclude, by definition of $\downarrow$ for '$+$', $(t'_p+t''_p){\downarrow}$ and hence $f(t_1,\ldots,t'_p+t''_p,\ldots,t_n){\downarrow}$ by application of θ.

(b) Suppose $p \in \mathit{neg}(f)$ and consider a closed instance $f(t_1,\ldots,t'_p+t''_p,\ldots,t_n) = f(t_1,\ldots,t'_p,\ldots,t_n)+f(t_1,\ldots,t''_p,\ldots,t_n)$ of a distributive law for p with respect to f. So $t_q \equiv \varepsilon$ for all $q \in \mathit{nonneg}(f)$. Recall that f is smooth. Hence, as $\mathit{act}(\rho) \subseteq \mathit{nonneg}(f)$ and $\mathit{act}(\rho) \neq \emptyset$, by definition, for every transition rule ρ for f, both $f(t_1,\ldots,t'_p+t''_p,\ldots,t_n)$ and $f(t_1,\ldots,t'_p,\ldots,t_n)+f(t_1,\ldots,t''_p,\ldots,t_n)$ have no transitions. Also $(t'_p+t''_p){\downarrow}$ iff $t'_p{\downarrow}$ or $t''_p{\downarrow}$. From this it follows that $f(t_1,\ldots,t'_p+t''_p,\ldots,t_n){\downarrow}$ iff $f(t_1,\ldots,t'_p,\ldots,t_n){\downarrow}$ or $f(t_1,\ldots,t''_p,\ldots,t_n){\downarrow}$, and therefore $f(t_1,\ldots,t'_p+t''_p,\ldots,t_n){\downarrow}$ iff $f(t_1,\ldots,t'_p,\ldots,t_n)+f(t_1,\ldots,t''_p,\ldots,t_n){\downarrow}$. $\square$

Note the observation that $\mathit{act}(\rho) \neq \emptyset$ which follows, for non-constant operations, directly from Definition 12 of the smoothness of an operation in the last paragraph of the proof of the lemma.

Next we consider the action laws. It is here that the notion of distinctivity comes into play. In short, distinctivity captures that for a source $f(t_1,\ldots,t_n)$, with f smooth and distinctive, at most one rule can apply. As can be seen from the proof sketch for the lemma, all conditions of Definition 16 regarding distinctivity are exploited.

Lemma 29 Let f be an n-ary smooth and distinctive operation of a *tagh*-transition system specification *TSS*. Then it holds that the action laws for the operation f are sound.

Proof Let ρ be a transition rule for f of the format (3). Let $f(t_1,\ldots,t_n) = a.C[t'_i,t'_j,t_\ell]$ be a closed instance of the action law (5) for the rule ρ for f. Hence $t_i \equiv a_i.t'_i$ for $i \in I$, $t_j \equiv \partial^1_{B_j}(t'_j)$ for $j \in J$ and $t_k \equiv \varepsilon$ for $k \in K$. Again we apply a first step analysis based on Lemma 3 to show that $f(t_1,\ldots,t_n)$ and $a.C[t'_i,t'_j,t_\ell]$ are bisimilar.

(i) Clearly, $f(t_1,\ldots,t_n) \xrightarrow{a} C[t'_i,t'_j,t_\ell]$ by application of ρ. This transition is matched by $a.C[t'_i,t'_j,t_\ell] \xrightarrow{a} C[t'_i,t'_j,t_\ell]$. Next we show, appealing to the distinctiveness of f, that $f(t_1,\ldots,t_n)$ admits no other transitions than the one based on ρ.

Suppose $f(t_1,\ldots,t_n) \xrightarrow{c} t$ via some rule ρ' for f of the format (3) with $\rho' \neq \rho$. First we derive that $\mathit{rank}(\rho') = \mathit{rank}(\rho)$ by falsification of the two cases

$\mathit{rank}(\rho) \succ \mathit{rank}(\rho')$ and $\mathit{rank}(\rho) \prec \mathit{rank}(\rho')$: (1) Assume $\mathit{rank}(\rho) \succ \mathit{rank}(\rho')$, then either $\mathit{pass}(\rho) \setminus \mathit{pass}(\rho') \neq \emptyset$ or $\mathit{act}(\rho) \cap \mathit{term}(\rho') \neq \emptyset$. In the first case we have, by distinctiveness of f (cf. the 2nd bullet of Definition 14), that $\mathit{act}(\rho) \cap \mathit{term}(\rho') \neq \emptyset$. Hence, in both cases, we can choose a position $q \in \mathit{act}(\rho) \cap \mathit{term}(\rho')$. But then we have $t_q \equiv a_q.t'_q$ as $q \in \mathit{act}(\rho)$ and $t_1\!\downarrow$ as $q \in \mathit{term}(\rho')$. Contradiction, $(a_q.t'_q)\not\downarrow$. (2) Assume $\mathit{rank}(\rho) \prec \mathit{rank}(\rho')$. Following a similar argument as above we can choose a position $q \in \mathit{act}(\rho') \cap \mathit{term}(\rho)$. But then we have $t_q \xrightarrow{a'_q} t'_q$ as $q \in \mathit{act}(\rho')$ and $t_q \equiv \varepsilon$ as $q \in \mathit{term}(\rho)$. Contradiction, ε admits no transitions. Since neither $\mathit{rank}(\rho) \succ \mathit{rank}(\rho')$ nor $\mathit{rank}(\rho) \prec \mathit{rank}(\rho')$ and the ranks of a smooth and distinctive operation are linearly ordered, we conclude that $\mathit{rank}(\rho) = \mathit{rank}(\rho')$ by distinctiveness of f (cf. the 1st bullet of Definition 16).

From $\mathit{rank}(\rho) = \mathit{rank}(\rho')$ we obtain $\mathit{act}(\rho) = \mathit{act}(\rho')$. If $\rho \neq \rho'$ we can choose, by distinctiveness of f (cf. the 2nd bullet of Definition 16), an index i such that $a_i \neq a'_i$. But then we have both $t_i \equiv a_i.t'_i$ and $t_i \xrightarrow{a'_i} t''_i$ for some term t''_i. Contradiction, $a_i.t_i$ admits no a'_i-transition. We conclude that ρ and ρ' must coincide and that $f(t_1, \ldots, t_n)$ only admits the transition based on the transition rule ρ.

(ii) The term $a.C[t'_i, t'_j, t_\ell]$ only has $a.C[t'_i, t'_j, t_\ell] \xrightarrow{a} C[t'_i, t'_j, t_\ell]$ as a transition, which is matched by $f(t_1, \ldots, t_n) \xrightarrow{a} C[t'_i, t'_j, t_\ell]$.

(iii) For every termination rule θ, we have $\mathit{act}(\rho) \cap \mathit{term}(\theta) \neq \emptyset$. Therefore, for each θ, $\exists p \in \mathit{act}(\rho) \cap \mathit{term}(\theta)$: $t_p \equiv a_p.t'_p$. Hence, by definition of $\downarrow$ for f, we have $f(t_1, \ldots, t_n)\not\downarrow$. Note also $a.C[t'_i, t'_j, t_\ell]\not\downarrow$ as prefixing has no termination rules. $\square$

The soundness of the deadlock laws is straightforward. The particular rank, for which a deadlock law is formulated, does not play a role here, but will become important for the head-normalization result (see Lemma 35) in the next section.

Lemma 30 Let f be an n-ary smooth and distinctive operation of a *tagh*-transition system specification *TSS*. Then it holds that the deadlock laws for the operation f are sound.

Proof Let R be a rank of f and let $f(t_1, \ldots, t_n) = \delta$ be a closed instance of a deadlock law for R. Hence $t_p \equiv \varepsilon$, $t_p \equiv \delta$ or $t_p \equiv a'_p.t'_p$ for some a'_p, t'_p for $p \in \mathit{act}(R)$ and for each rule ρ for f of the format (3) one of the following cases holds: (1) $\exists i \in \mathit{act}(\rho)$: $t_i \equiv \delta$ or $t_i \equiv a'_i.t'_i$ and $a'_i \neq a_i$, (2) $\exists j \in \mathit{neg}(\rho)$: $t_j \equiv b'_j.t'_j$ or $t_j \equiv t''_j + b.t'_j$ for some t''_j, b, t'_j with $b \in B_j$, (3) $\exists k \in \mathit{term}(\rho)$: $t_k \equiv \delta$ or $t_k \equiv a'_k.t'_k$. It follows that for each rule ρ for f of the format (3): (1) $\exists i \in \mathit{act}(\rho)$: $t_i \stackrel{a_i}{\not\to}$, (2) $\exists j \in \mathit{neg}(\rho)$: $t_j \xrightarrow{b} t'_j$ for some action $b \in B_j$ and some term t'_j, or (3) $\exists k \in \mathit{term}(\rho)$: $t_k \not\downarrow$.

We conclude that $f(t_1, \ldots, t_n)$ has no transitions, just as δ does. Moreover, by definition of the deadlock law, we have, for each termination rule θ, $\exists p \in \mathit{term}(\theta)$: $t_p \equiv \delta$ or $t_p \equiv a'_p.t'_p$. Hence $f(t_1, \ldots, t_n)\not\downarrow$. By definition of $\downarrow$ for δ, we also have $\delta\not\downarrow$. $\square$

The proof of the last soundness lemma regarding a smooth and distinctive operation makes use of the fact that for a transition rule ρ of the format (3), it holds that $\mathit{act}(\rho) \cap \mathit{term}(\theta) \neq \emptyset$. So, the termination rule θ guarantees the term ε at a position where the transition rule ρ demands an action.

Lemma 31 Let f be an n-ary smooth and distinctive operation of a *tagh*-transition system specification *TSS*. Then it holds that the termination laws for the operation f are sound.

Proof Let $f(t_1, \ldots, t_n)$ be a closed instance of a termination law for a termination rule θ for f. Hence $t_p \equiv \varepsilon$ for all $p \in \mathit{term}(\theta)$. For all rules ρ for f we have that $\mathit{act}(\rho) \cap \mathit{term}(\theta) \neq \emptyset$ by distinctiveness of f. So $f(t_1, \ldots, t_n)$ has no transitions, just as ε does. Moreover, both $f(t_1, \ldots, t_n)\downarrow$, since $\forall p \in \mathit{term}(\theta)$: $t_p\downarrow$, and $\varepsilon\downarrow$ by definition. $\square$

The next result concerns the soundness of the distinctivity law for a smooth but non-distinctive operation. The construction and its proof are a modest extension of the corresponding lemma of [ABV94]. As only extra we need for the termination condition of Definition 2 that a sum can terminate iff one its summands can terminate, a fact which directly follows from the termination rules for '+' in TSS_∂.

Lemma 32 For an n-ary smooth operation f in a *tagh*-transition system specification *TSS*, there exists a disjoint extension TSS' with smooth and distinctive n-ary operations f_1 through f_s, say, such that $f(z_1, \ldots, z_n) = f_1(z_1, \ldots, z_n) + \cdots + f_s(z_1, \ldots, z_n)$ is sound for bisimulation on the transition system space generated by TSS'.

Proof Start, as in [ABV94], from a partitioning $R_1, \ldots, R_s$ of the rules for f in TS such that f is smooth and distinctive with respect to each of the parts. Introduce, for each part R_r, a fresh n-ary operation f_r with as its rules copies of the rules in R_r with f replaced by f_r and add this to *TSS*. Then f_r is a smooth operation. Moreover, we have that $f(t_1, \ldots, t_n) \xrightarrow{a} t$ iff $f_r(t_1, \ldots, t_n) \xrightarrow{a} t$ for some $r \in \{1, \ldots, s\}$, and $f(t_1, \ldots, t_n)\downarrow$ iff $f_r(t_1, \ldots, t_n)\downarrow$ for some $r \in \{1, \ldots, s\}$. Hence it follows that $f((z_1, \ldots, z_n)$ and $f_1(z_1, \ldots, z_n) + \cdots + f_s(z_1, \ldots, z_n)$ are bisimilar. $\square$

The soundness proof of the final building block, viz. the transition from a non-smooth operation to a smooth one, is based on the construction of Definition 24. For simplicity we suppress the issue of absence of active transitions. Two points remain: (i) to establish the number of copies that should be introduced for each argument, and (ii) to verify that the two operations admit the same transitions.

Lemma 33 Let f be a non-smooth n-ary operation of a transition system specification *TSS*. Then there exists a disjoint extension *TSS*′ of *TSS* with a smooth m-ary operation f' and an equation $f(z_1, \ldots, z_n) = f'(\zeta_1, \ldots, \zeta_m)$ with z_p, for $p \in \{1, \ldots, n\}$, all different and $\zeta_q \in \{z_1, \ldots, z_n\}$, for $q \in \{1, \ldots, m\}$ that is sound for bisimulation on the transition system space generated by *TSS*′.

Proof The proof follows exactly the reasoning of [ABV94]: First we have to establish the number of 'copies' for each argument, using the so-called barb-factor. Then, we need a technical result (Lemma 4.12 of [ABV94]) to show that the copies will generate the same terms as their sources, i.e., that $f(z_1, \ldots, z_n) \xrightarrow{a} t \iff f'(\zeta_1, \ldots, \zeta_m) \xrightarrow{a} t$. Finally we have to check that the termination condition for bisimulation with explicit termination holds. □

By now we have addressed all the laws raised in the previous section. Concatenation of the above lemmata now yields the desired soundness result.

Theorem 34 *Let TSS be a transition system specification in tagh-format with generated transition system specification TSS′ and generated equational theory ET'. Then the theory ET' is sound with respect to the transition system space generated by TSS′ modulo bisimulation.* □

Note that in passing, we have in the results above made use of all format prescriptions and additional bulleted requirements of Definitions 12, 14 and 16. Also note that, as all extensions of *TSS* as yielded by the procedure of Definition 26 are disjoint, if follows that the theory ET' is sound with respect to TS modulo bisimulation as well (see Section 2).

5 Completeness

In this section we show, for a *tagh*-transition system specification *TSS*, the completeness of the generated set of equations ET' for the transition system space generated by the extended specification *TSS*′ modulo bisimulation. We follow the outline as provided in [ABV94].

The first result concerns head-normalization of the generated equational theory ET' and will be used as a tool to find a 'projection' t'/σ^n (see below) of a term t over the signature $\{\varepsilon, \delta, a.\ , +\}$ in the process algebra, such that $ET' \vdash t'/\sigma^n = t$. The proof of the result requires a detailed case analysis that exploits the full machinery of *handle*, *rank* and the ordering $\succcurlyeq$ on transition rules.

Lemma 35 Let *TSS* be a transition system specification in *tagh*-format with generated transition system specification *TSS*$'$ and equational theory ET'. Then the theory ET' is head-normalizing for terms over *TSS*$'$.

Proof In the presence of the smoothening and distinctivity laws, it suffices to show that for any n-ary smooth and distinctive operation f and closed terms $t_1, \dots, t_n$ in head-normal form, we have that $ET' \vdash f(t_1, \dots, t_n) = t$ for some closed term t in head normal-form (cf. Definition 9). We elaborate a detailed case analysis:

(1) Assume that there is $m \in \mathit{nonneg}(f)$ such that t_m is a sum, has '+' as the main operation. Choose the index m maximal in rank such that t_m is a sum, say $t_m \equiv t'_m + t''_m$. We distinguish two subcases:

(a) $[\forall p \in \mathit{handle}(m)\colon t_p \equiv \varepsilon]$ Simply define $t \equiv f(t_1, \dots, t'_m, \dots, t_n) + f(t_1, \dots, t''_m, \dots, t_n)$ and apply the distributive law for m to obtain $f(t_1, \dots, t_n) = t$.

(b) $[\exists p \in \mathit{handle}(m)\colon t_p \equiv \delta$ or $t_p \equiv a_p.t'_p$ for some $a_p, t'_p]$ Note that if $p \in \mathit{handle}(m)$ then t_p cannot be a sum, since $p \in \mathit{handle}(m)$ implies $\mathit{rank}(p) \succ \mathit{rank}(m)$ and m was chosen to be maximal in rank with t_m a sum.

Suppose $f(t_1, \dots, t_n) \xrightarrow{a} t'$ for some term t' and rule ρ. We either have $\mathit{term}(\rho) \supsetneq \mathit{term}(m)$ or $\mathit{term}(\rho) = \mathit{term}(m)$ or $\mathit{term}(\rho) \subsetneq \mathit{term}(m)$. By the assumption on the position $p \in \mathit{handle}(m) = \mathit{term}(m)$ we then have $\mathit{term}(\rho) \subsetneq \mathit{term}(m)$, so $\mathit{rank}(\rho) \succ \mathit{rank}(m)$. If $i \in \mathit{act}(\rho)$, then $\mathit{rank}(i) \succ \mathit{rank}(m)$, so t_i cannot be a sum by maximality of m in rank, and, hence, $t_i \equiv a_i.t'_i$. For $j \in \mathit{neg}(\rho)$ we have $t_j \stackrel{b}{\nrightarrow}$ for $b \in B_j$, from which it follows that $t_j = \partial^1_{B_j}(t_j)$. If $k \in \mathit{term}(\rho)$, then $\mathit{rank}(k) \succ \mathit{rank}(m)$ by distinctiveness of f, so t_k cannot be a sum, again by maximality of m in rank, and therefore $t_k \equiv \varepsilon$. Now, put $t \equiv a.C[t'_i, t_j, t_\ell]$ and apply the action law for ρ to obtain $f(t_1, \dots, t_n) = t$.

Suppose $f(t_1, \dots, t_n)$ admits no rules. For each rule ρ for f such that $\mathit{rank}(\rho) \succ \mathit{rank}(m)$ we have that t_q cannot be a sum for $q \in \mathit{act}(\rho) \cup \mathit{term}(\rho) \subseteq \mathit{nonneg}(f)$. As such ρ does not match $f(t_1, \dots, t_n)$ it holds that (1) $\exists i \in \mathit{act}(\rho)\colon t_i \stackrel{a_i}{\nrightarrow}$, (2) $\exists j \in \mathit{neg}(\rho)\colon t_j \xrightarrow{b} t'_j$ for some action $b \in B_j$ and some term t'_j, or (3) $\exists k \in \mathit{term}(\rho)\colon t_k {\not\downarrow}$. From this we derive that (1) $\exists i \in \mathit{act}(\rho)\colon t_i \equiv \varepsilon$, $t_i \equiv \delta$ or $t_i = a'_i.t'_i$ with $a'_i \neq a_i$, (2) $\exists j \in \mathit{neg}(\rho)\colon t_j \equiv b.t'_j$ or $t_j \equiv t''_j + b.t'_j$ for some action $b \in B_j$ and some term t'_j, or (3) $\exists k \in \mathit{term}(\rho)$: $t_k \equiv \delta$ or $t_k \equiv a'_k.t'_k$.

For each rule ρ for f such that $\mathit{rank}(\rho) \preccurlyeq \mathit{rank}(m)$ we have that $\mathit{term}(\rho) \supseteq \mathit{handle}(m)$. Since, by assumption, $\exists p \in \mathit{handle}(m)$: $t_p \equiv \delta$ or $t_p \equiv a'_p.t'_p$, it follows that $\exists k \in \mathit{term}(\rho)$: $t_k \equiv \delta$ or $t_k \equiv a'_k.t'_k$. If, for all termination rules θ, $\exists p \in \mathit{term}(\theta)$: $t_p \not\equiv \varepsilon$, put $t \equiv \delta$ and apply the corresponding deadlock law for $\mathit{rank}(m)$. If not, there exists a termination rule θ for f such that $\forall p \in \mathit{term}(\theta)$: $t_p \equiv \varepsilon$. Put $t \equiv \varepsilon$ and apply the termination law for θ. In both cases we obtain $f(t_1, \ldots, t_n) = t$.

(2) Assume that for all $m \in \mathit{nonneg}(f)$ we have that $t_m \equiv \varepsilon$, $t_m \equiv \delta$ or $t_m \equiv a'_m.t'_m$ for some a'_m, t'_m. We distinguish three subcases:
  (i) [$f(t_1, \ldots, t_n)$ has a transition] Suppose $f(t_1, \ldots, t_n) \xrightarrow{a} C[t'_i, t_j, t_\ell]$ for some rule ρ of the form (3). Put $t \equiv a.C[t'_i, t_j, t_\ell]$ and apply the action law for ρ to obtain $f(t_1, \ldots, t_n) = t$.
  (ii) [$\forall p \in \mathit{term}(\theta)$: $t_m \equiv \varepsilon$ for some termination rule θ]. Put $t \equiv \varepsilon$ and apply the corresponding termination law for θ to obtain $f(t_1, \ldots, t_n) = t$.
  (iii) [$f(t_1, \ldots, t_n)$ admits no transition rule and, for no termination rule θ, $\forall p \in \mathit{term}(\theta)$: $t_m \equiv \varepsilon$] If $\forall m \in \mathit{nonneg}(f)$: $t_m \equiv \varepsilon$ and $\exists j \in \mathit{neg}(f)$: t_j is a sum, apply the distributive law for j. If not, put $t \equiv \delta$ and apply the deadlock law for a rank of f with $\mathit{pass}(R) = \emptyset$ (which, by smoothness of f, exists as every position p is negative or becomes active or terminating eventually). In both cases we obtain $f(t_1, \ldots, t_n) = t$. $\square$

Having the head-normalization result in place, we can conclude, using standard arguments, the completeness of the generated theory for finite processes. However, in order to deal with infinite behaviour, we need, in line with [ABV94], some extra machinery. First, we introduce a syntactic version of the Approximation Induction Principle (cf. Lemma 36). Next, we show that all 'projections' can be represented by a term for the basic transition system specification TSS_∂ (cf. Lemma 37). The results are then combined (see Theorem 38) to obtain the announced completeness result.

Let *TSS* be a *tagh*-transition system specification. The transition system specification $\mathit{TSS}_{/}$ is the disjoint extension of *TSS* and TSS_∂ with only one binary operation '/', referred to as the hourglass operation. This hourglass operation is defined by the following rules:

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

Let σ be an arbitrary action from Act, that we think as indicating a sandgrain for the hourglass. For $n \in \mathbb{N}$, the term σ^n is defined by $\sigma^0 \equiv \delta$, $\sigma^{n+1} \equiv \sigma.\sigma^n$.

Clearly, for two closed terms t_1 and t_2 that are bisimilar transition systems as generated by *TSS*, we have that t_1/σ^n and t_2/σ^n are bisimilar transition systems as generated by *TSS* extended with $TSS_{/}$. The Approximation Induction Principle, *AIP* for short, can now be reformulated in terms of the hourglass and sandgrains:

$$\frac{x/\sigma^n = y/\sigma^n \quad (\forall n \in \mathbb{N})}{x = y}$$

We then have the following fundamental result, based on the finite branching of a transition system generated by a *tagh*-transition system specification.

Lemma 36 Let *TSS* be a disjoint extension of $TSS_{/}$. Then on the transition system space generated by *TSS*, *AIP* holds, i.e., if, for closed terms t_1, t_2 over *TSS*, it holds that $\forall n \in \mathbb{N}$: $t_1/\sigma^n \sim t_2/\sigma^n$, then also $t_1 \sim t_2$.

Proof See [Gla87]. □

It is here, that finite branching of the transition system space is used. Finite branching is guaranteed as a *tagh*-transition system specification and the set of actions are finite. The requirement of finite branching can be loosened, but then things become much more complicated.

The hourglass operation '/', as can be directly seen from its rules, is smooth and distinctive. Therefore, by the results of the previous section, we have that, amongst others, the following equations hold (with respect to any disjoint extension of $TSS_{/}$):

$$\begin{array}{ll} (x_1 + x_2)/y = (x_1/y) + (x_2/y) & \qquad \delta/y = \delta \\ (a.x')/(b.y') = a.(x'/y') & \qquad \varepsilon/y = \varepsilon \end{array}$$

Lemma 37 Let *TSS* be a *tagh*-transition system specification and TSS' the disjoint extension of $TSS_{/}$ with accompanying equational theory ET' generated by the procedure in Definition 26. Then it holds for any closed term t' for TSS' and any $n \in \mathbb{N}$, that there exists a closed term t in the signature of TSS_0 such that $ET' \vdash t'/\sigma^n = t$ and $t'/\sigma^n \sim t$ in the generated transition system space.

Proof The proof goes, as in [ABV94], by induction on n using the head-normalization result Lemma 35 and the equations for the operation '/' above. □

We are now in a position to provide the completeness result for the equations synthesized by the generation procedure. The proof of the theorem below is similar to the proof presented in [ABV94] for the *GSOS*-format. It is included here to show the reader the interplay of the various results presented above.

Theorem 38 *Let TSS be a tagh-transition system specification. Let TSS' be the disjoint extension of $TSS_{/}$ and the generated extension of TSS. Let ET' be the generated equational theory. Then ET' and AIP are sound and complete for the generated transition system space modulo bisimulation.*

Proof If $ET', AIP \vdash t' = t''$ for two closed terms t', t'' over TSS', then it follows from the soundness results of Section 4 and Lemma 36 that t' and t'' have also bisimilar transition systems.

Suppose t', t'' are closed and bisimilar terms over TSS'. Then we have that t'/σ^n and t''/σ^n are bisimilar transition systems, for all $n \in \mathbb{N}$. Now, by virtue of *AIP*, it suffices to show that $ET' \vdash t'/\sigma^n = t''/\sigma^n$ for each n in $\mathbb{N}$. So, pick $n \in \mathbb{N}$. Choose, using Lemma 37 two closed terms t_1, t_2 such that $ET' \vdash t'/\sigma^n = t_1$ and $ET' \vdash t''/\sigma^n = t_2$. From the soundness of ET' we derive that t'/σ^n and t_1 are bisimilar and that t''/σ^n and t_2 are bisimilar. Hence, t_1 and t_2 are bisimilar. Note that TSS' is a disjoint extension of TSS_∂. We thus obtain that t_1 and t_2 are bisimilar as transition systems generated by TSS_∂. By the completeness result for TSS_∂, Lemma 8, it follows that $ET_\partial \vdash t_1 = t_2$ and, a fortiori, $ET' \vdash t_1 = t_2$. We conclude that $ET' \vdash t'/\sigma^n = t''/\sigma^n$, as was to be shown. □

6 Concluding remarks

We have introduced the *tagh*-format for structured operational semantics. The *tagh*-format enhances the well-known *GSOS*-format with explicit termination. In addition, the format allows for a finer distinction between the modes of the argument (viz. active, negative, terminating, passive). The method of automatic generation of axiomatizations as developed by Aceto, Bloom and Vaandrager for *GSOS* is extended for the case of *tagh*. We have shown that for a transition system specification in *tagh*-format the synthesized theory is sound and complete for the induced transition system space modulo bisimulation. Examples illustrate the technique and indicate the strength of the approach. The resulting laws are equal or close to hand-crafted axiomatizations.

Many other examples than the ones mentioned have been examined already. E.g., the projection operator (related to the hourglass operator above), renaming operator, encapsulation, restriction, state operator, generalized state operator and process creation operator can be treated within the framework of the *tagh*-format. Following the practical thread, our aim is to experiment with more extensive transition systems and to investigate the impact of the axiomatization method, for example for timed transition systems. A theoretical issue here is the adaptation of the techniques for the *tagh*-format to deal with implicit termination of the form $x \xrightarrow{a} \surd$, a format at present also often used within process algebra. Furthermore, the approach presented here should be extended to deal with transition system specifications and process algebras comprising quantitative information, such as timed and probabilistic systems.

In order to pave the way for mechanized axiomatizations of more extensive, larger transition system specifications than the ones discussed in this paper, initial efforts have been spent on tool support. Michiel Tas, in a thesis project conducted at Technische Universiteit Eindhoven has implemented the algorithm set out in Definition 26 including a ranking strategy [Tas03]. The user can either provide a transition system specification to the tool as text or enter or modify a transition system specification through a GUI. The resulting equational theory is output in text-format or as LaTeX code dependent on the users choice. Various examples have been checked with the tool. Amongst others, the tool was capable of generating the equational theory for the functional core of the χ-language of [BK02] consisting of no less than 44 transition and termination rules for its fundamental operators. The χ-language is used by the Mechanical Engineering Department of Technische Universiteit Eindhoven for the modeling and analysis of practical equipment ranging from modestly sized clinical chemical analyzers to complex wafersteppers.

Another, theoretically important question concerns the application of the *tagh*-format in the setting of metric semantics and co-induction (see, for example, [BV96,Rut00]). In this paper we have focused on transition systems and their axiomatizations. Another view is to consider transition systems and denotational models (cf. [Rut90,AI96,TP97]). We believe, having the correspondence of the syntactic ε with the empty semantical process p_ε of metric domain equations, it should be feasible to automatically construct higher-order or co-inductive definitions for semantical operators and a denotational semantics that is correct with respect to a transition system in *tagh*-format.

References

[ABV94] L. Aceto, B. Bloom, and F.W. Vaandrager. Turning SOS rules into equations. *Information and Computation*, 111:1–52, 1994.

[AFV01] L. Aceto, W.J. Fokkink, and C. Verhoef. Structural operational semantics. In J.A. Bergstra, A. Ponse, and S.A. Smolka, editors, *Handbook of Process Algebra*, pages 197–292. Elsevier Science, 2001.

[AH92] L. Aceto and M. Hennessy. Termination, deadlock and divergence. *Journal of the ACM*, 39:147–187, 1992.

[AI96] L. Aceto and A. Ingólfsdóttir. CPO models for compact GSOS languages. *Information and Computation*, 129:107–141, 1996.

[Bae03] J.C.M. Baeten. Embedding untimed into timed process algebra: The case for explicit termination. *Mathematical Structures in Computer Science*, 13:589–618, 2003.

[BBR03] J.C.M. Baeten, T. Basten, and M.A. Reniers. *Algebra of Communicating Processes*. Cambridge University Press, 2003. To appear.

[BBM01] J.C.M. Baeten, H.M.A. van Beek, and S. Mauw. Specifying internet applications with DiCons. In *Proc. ACM Symp. on Applied Computing*, pages 576–584, Las Vegas, 2001. ACM.

[BB00] J.C.M. Baeten and J.A. Bergstra. Mode transfer in process algebra. Technical Report CSR 00-01, Division of Computer Science, Technische Universiteit Eindhoven, 2000.

[BBK86] J.C.M. Baeten, J.A. Bergstra, and J.W. Klop. Syntax and defining equations for an interrupt mechanism in process algebra. *Fundamenta Informaticae*, IX:127–168, 1986.

[BV95] J.C.M. Baeten and C. Verhoef. Concrete process algebra. In S. Abramsky, D.M. Gabbay, and T.S.E. Maibaum, editors, *Handbook of Logic in Computer Science*, volume 4, Syntactical Methods, pages 149–268. Oxford University Press, 1995.

[BV02] J.C.M. Baeten and E.P. de Vink. Axiomatizing gsos with termination (extended abstract). In H. Alt and A. Ferreira, editors, *Proc. STACS'02*, pages 583–595. LNCS 2295, 2002.

[BW90] J.C.M. Baeten and W.P. Weijland. *Process Algebra*, volume 18 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 1990.

[BV96] J.W. de Bakker and E.P. de Vink. *Control Flow Semantics*. Foundations of Computing Series. MIT Press, 1996.

[BR97] D.A. van Beek and J.E. Rooda. Specification and simulation of industrial systems using an executable mathematical specification language. In *Proc. 15th IMACS World Congress, Vol. 2 Numerical Mathematics*, pages 721–726, Berlin, 1997.

[BK84] J.A. Bergstra and J.W. Klop. Process algebra for synchronous communication. *Information and Control*, 60:109–137, 1984.

[BIM95] B. Bloom, S. Istrail, and A.R. Meyer. Bisimulation can't be traced. *Journal of the ACM*, 42:232–268, 1995. Preliminary version in Proc. POPL'88.

[BK02] V. Bos and J.T.T. Kleijn. *Formal specification and analysis of industrial systems*. PhD thesis, Technische Universiteit Eindhoven, 2002.

[Bri89] E. Brinksma, editor. *Information Processing Systems, Open Systems Interconnection, LOTOS – A Formal Description Technique Based on the Temporal Ordering of Observational Behaviour.* ISO Standard IS–8807, 1989.

[BHR84] S.D. Brookes, C.A.R. Hoare, and A.W. Roscoe. A theory of Communicating Sequential Processes. *Journal of the ACM*, 31(3):560–599, 1984.

[EHS97] J. Ellsberger, D. Hogrefe, and A. Sarma. *SDL: Formal Object-Oriented Language for Communicating Systems.* Prentice Hall, 1997.

[Fok00] W.J. Fokkink. *Introduction to Process Algebra.* Texts in Theoretical Computer Science, An EATCS Series. Springer, 2000.

[Gla87] R.J. van Glabbeek. Bounded nondeterminism and the approximation induction principle in process algebra. In F.J. Brandenburg, G. Vidal-Naquet, and M. Wirsing, editors, *Proc. STACS'87*, pages 336–347. LNCS 247, 1987.

[Mil80] R. Milner. *A Calculus of Communicating Systems.* LNCS 92, 1980.

[Mil89] R. Milner. *Communication and Concurrency.* Prentice-Hall International, 1989.

[Rut90] J.J.M.M. Rutten. Deriving denotational models for bisimulation from structured operational semantics. In M. Broy and C.B. Jones, editors, *Proc. IFIP Working Group 2.2/2.3 Working Conference*, pages 155–177, 1990.

[Rut00] J.J.M.M. Rutten. Universal coalgebra: A theory of systems. *Theoretical Computer Science*, 249:3–80, 2000.

[Tas03] M.A. Tas. Synthesis of theories for specifications in tagh-format. Technische Universiteit Eindhoven, 2003.

[TP97] D. Turi and G. Plotkin. Towards a mathematical operational semantics. In *Proc. LICS'97*, pages 280–291. IEEE, 1997.

[Uli95] I. Ulidowski. Axiomatisations of weak equivalences for De Simone languages. In I. Lee and S. Smolka, editors, *Proc. CONCUR'95*, pages 219–233. LNCS 962, 1995.

[Uli00] I. Ulidowski. Finite axiom systems for testing preorder and De Simone process languages. *Theoretical Computer Science*, 239:97–139, 2000.

[Ver95] C. Verhoef. A congruence theorem for structured operational semantics with predicates and negative premises. *Nordic Journal of Computing*, 2:274–302, 1995.

[Vra97] J.L.M. Vrancken. The algebra of communicating processes with empty process. *Theoretical Computer Science*, 177:287–328, 1997.