Axiomatizing GSOS with Termination

J.C.M. Baeten[1] and E.P. de Vink[1,2]

[1] Department of Mathematics and Computer Science, Eindhoven University of Technology, P.O. Box 513, 5600 MB Eindhoven, The Netherlands, {josb,evink}@win.tue.nl
[2] 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 allows besides active and negative premises as in *GSOS*, also for, what is called 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 system specifications. 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*. The examples provided indicate a significant improvement over the mechanical axiomatization techniques known so far.

1 Introduction

It has become very popular in the concurrency community to define various process operators by means of Plotkin-style operational rules. 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.

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 have infinitely many states. In these cases, an approach using theorem provers or deploying 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, we have seen no application of the theory. The reader may wonder why this is so.

H. Alt and A. Ferreira (Eds.): STACS 2002, LNCS 2285, pp. 583–595, 2002.

In our opinion, this is due to the limited process algebraic basis employed in [ABV94]; in particular, termination and deadlock are identified. Any language, both programming and specification languages, involving some form of parallel composition will know 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, subsequently, 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, in our opinion an unsatisfactory solution. Second, we can use *implicit termination* as in ACP, where successful termination is implicitly "tacked onto" 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. 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 can be extended in order to deal also with implicit termination, but leave this as future research. Starting from the *GSOS*-format (cf. [BIM95]), we extend it with termination to obtain the *tagh*-format (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 almost optimal, intuitively understandable, and are sound and complete for the model of transition systems modulo bisimulation.

2 Preliminaries

We assume the reader to be familiar with the standard notions and examples of process algebra (cf., e.g., [BW90,Fok00]). Below we present the transition system specification for the basic process language with explicit termination ε, deadlock δ (which has no rules), a prefixing operation '$a.$ ' for every a taken from the finite alphabet of actions Act, nondeterministic choice '$+$' and unary one-step restriction operations ∂_B^1 for every subset $B \subseteq Act$. The expression $\partial_B^1(t)$ 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. For the termination predicate '$\downarrow$', we use the postfix notation $t\downarrow$ meaning that the term t has an option to terminate immediately.

Definition 1

(a) The transition system specification for the transition system TSS^1_∂ consists of the following transition and termination rules:

$$a.x \xrightarrow{a} x \qquad \frac{x \xrightarrow{a} x'}{x+y \xrightarrow{a} x'} \qquad \frac{y \xrightarrow{a} y'}{x+y \xrightarrow{a} y'} \qquad \frac{x \xrightarrow{a} x'}{\partial^1_B(x) \xrightarrow{a} x'}\ (a \notin B)$$

$$\varepsilon\!\downarrow \qquad \frac{x\!\downarrow}{(x+y)\!\downarrow} \qquad \frac{y\!\downarrow}{(x+y)\!\downarrow} \qquad \frac{x\!\downarrow}{\partial^1_B(x)\!\downarrow}$$

(b) The equational theory ET^1_∂ consists of the following equations:

$$\begin{array}{lll} x+y = y+x & \partial^1_B(x+y) = \partial^1_B(x) + \partial^1_B(y) & \partial^1_B(\delta) = \delta \\ (x+y)+z = x+(y+x) & \partial^1_B(a.x) = a.x \ \ \textit{if } a \notin B & \partial^1_B(\varepsilon) = \varepsilon \\ x+x = x & \partial^1_B(a.x) = \delta \ \ \textit{if } a \in B & \\ x+\delta = x & & \end{array}$$

We have the standard notion of strong bisimulation with predicates, in our set-up in the form of a termination condition.

Definition 2 *A bisimulation relation R for a transition system TSS is a binary relation for closed terms over TSS such that whenever $t_1 R t_2$ it holds that (i) $t_1 \xrightarrow{a} t_1' \implies \exists t_2' \colon t_2 \xrightarrow{a} t_2' \wedge t_1' R t_2'$, (ii) $t_2 \xrightarrow{a} t_2' \implies \exists t_1' \colon t_1 \xrightarrow{a} t_1' \wedge t_1' R t_2'$, (iii) $t_1\!\downarrow \iff t_2\!\downarrow$. Two terms t_1, t_2 are bisimilar with respect to TSS if there exists a bisimulation relation R for TSS with $t_1 R t_2$, notation: $t_1 \sim_{TSS} t_2$ or just $t_1 \sim t_2$.*

Below we will use $t_1 \equiv t_2$ to denote 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 index sets K, L and M.

3 Generating Equations for the *tagh*-Format

In this section we introduce the *tagh*-format for transition system specifications. Here, the acronym *tagh* stands for *termination and GSOS hybrid*. It enhances the *GSOS*-format as introduced in [BIM95] with a notion of explicit termination. We provide a general procedure to obtain, for each transition system specification in *tagh*-format, a disjoint extension TSS' and an equational theory ET'. It holds that ET' is sound and complete for TSS'-bisimulation (cf. [BV01]). As the transition system specification TSS' is a disjoint extension of the transition system specification 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 TSS-bisimulation.

Definition 3

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

$$\frac{\{\, x_i \xrightarrow{a_{ip}} y_{ip} \mid i \in I, p \in P_i \,\} \quad \{\, x_j \overset{b}{\nrightarrow} \mid j \in J, b \in B_j \,\} \quad \{\, x_k{\downarrow} \mid k \in K \,\}}{f(x_1, \ldots, x_n) \xrightarrow{a} C[x_m, y_{ip} \mid m \in \{1, \ldots, n\}, i \in I, p \in P_i]} \quad (1)$$

with $I, J, K \subseteq \{1, \ldots, n\}$, for $i \in I$, P_i a nonempty finite index set, for $j \in J$, B_j a finite (possibly empty) set of actions from Act, and, x_m, y_{ip}, for $m \in \{1, \ldots, n\}, i \in I, p \in P_i$, pairwise distinct variables, that are the only variables that may occur in the context C.

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

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

with $x_1, \ldots, x_n$ pairwise distinct variables and the index set $K \subseteq \{1, \ldots, n\}$.

(c) A tagh-transition system specification is a transition system specification where any operation f different from 'ε', 'δ', '$a.$ ', '$+$' and '∂_B^1' has tagh-transition rules and/or tagh-termination rules only.

In the context of a transition rule ρ of the format (1) we use $act(\rho)$, $neg(\rho)$, $term(\rho)$, $pass(\rho)$ to denote the index sets I, J, K, L, respectively, where $L = \{1, \ldots, n\} \setminus (I \cup J \cup K)$. For a 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 the target. Occasionally we will write $t{\not\downarrow}$ if *not* $t{\downarrow}$, i.e., t cannot terminate immediately.

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. We therefore introduce (cf. [ABV94]) a more restricted format, called smooth. Regarding 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 4 *Let TSS be a tagh-transition system specification.*

(a) A transition rule ρ in TSS for an n-ary operation $f \in Sig$ is smooth *if it is of the format*

$$\frac{\{\, x_i \xrightarrow{a_i} y_i \mid i \in I \,\} \quad \{\, x_j \overset{b}{\nrightarrow} \mid j \in J, b \in B_j \,\} \quad \{\, x_k{\downarrow} \mid k \in K \,\}}{f(x_1, \ldots, x_n) \xrightarrow{a} C[y_i, x_j, x_\ell \mid i \in I, j \in J, \ell \in L]} \quad (3)$$

where the index sets I, J, K, L *form a partition of* $\{1, \ldots, n\}$, $I \neq \emptyset$, $B_j \subseteq Act$ *a finite (possibly empty) subset of actions, and, where in the target* $C[y_i, x_j, x_\ell]$ *only variables amongst* $\{y_i \mid i \in I\}$, $\{x_p \mid p \in J \cup L\}$ *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.*

(b) The rank of a rule ρ *is the 4-tuple* $\langle pass(\rho), act(\rho), term(\rho), neg(\rho)\rangle$, *notation* $rank(\rho)$. *For two rules* ρ, ρ' *for an* n*-ary operation* f *we say that* $rank(\rho) \succcurlyeq rank(\rho')$ *iff*

- $neg(\rho) = neg(\rho')$, $pass(\rho) \supseteq pass(\rho')$ *and* $term(\rho) \subseteq term(\rho')$, *and*
- $pass(\rho) \neq pass(\rho') \implies act(\rho) \cap term(\rho') \neq \emptyset$.

(c) A smooth n*-ary operation* f *is called* smooth *and* distinctive *with respect to TSS if*

- *the set* $\{rank(\rho) \mid \rho$ *a transition rule for* f *in TSS*$\}$ *is totally ordered by the ordering '*$\succcurlyeq$*' introduced in part (b);*
- *for any two distinct rules* ρ, ρ' *of the form (3) with* $rank(\rho) = rank(\rho')$ *there exists an index* $i \in act(\rho) = act(\rho')$ *such that* $a_i \neq a'_i$.
- *for each termination rule* θ *and each transition rule* ρ *for* f *in TSS it holds that* $term(\theta) \cap act(\rho) \neq \emptyset$.

For such an 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 TSS.*

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 $\rho \succcurlyeq \rho'$. The ordering on $\succcurlyeq$ then demands that a passive position in ρ' must be passive in ρ as well and that a terminating position in ρ must be terminating in ρ' as well. 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. 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 $0 \leq k < \ell \leq m$ it holds that $p \in pass(\rho_i)$ for $1 \leq i \leq k$, $p \in act(\rho_i)$ for $k < i \leq \ell$ and $p \in term(\rho_i)$ 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 from passive, via active, to terminating (but, possibly, p doesn't start out as passive 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: If ρ and ρ' are two distinct rules for f, we either have $rank(\rho) = rank(\rho')$ or, without loss of generality, $rank(\rho) \succ rank(\rho')$. From the requirements of Definition 4c 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 $rank(\rho) \succ rank(\rho')$ that $act(\rho) \cap term(\rho') \neq \emptyset$. So, for some $i \in \{1, \ldots, n\}$ we have $t_i \equiv a.t'$ 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'.

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.

Examples 5 *The binary operation ';' of sequential composition comes equipped, in the set-up with explicit termination, with two transition rules and one termination rule:*

$$(Seq_1)\ \frac{x \xrightarrow{a} x'}{x;y \xrightarrow{a} x';y} \qquad (Seq_2)\ \frac{x\downarrow \quad y \xrightarrow{a} y'}{x;y \xrightarrow{a} y'} \qquad (Seq_\varepsilon)\ \frac{x\downarrow \quad y\downarrow}{(x;y)\downarrow}$$

The binary operation '$\between$', usually referred to as leftmerge, has one transition rule and one termination rule:

$$(Leftmerge_1)\ \frac{x \xrightarrow{a} x'}{x \between y \xrightarrow{a} x' \parallel y} \qquad (Leftmerge_\varepsilon)\ \frac{x\downarrow \quad y\downarrow}{(x \between y)\downarrow}$$

We have that both ';' and '$\between$' (contrasting [ABV94]) are smooth and distinctive operations.

Note that, in fact, in the above examples we have transition schemes for (Seq_1), (Seq_2) and $(Leftmerge_1)$ rather than transition rules, as we have transition rules (Seq_1), (Seq_2) and $(Leftmerge_1)$, respectively, for each action $a \in Act$.

Before we are ready to describe the axioms generated for a smooth and distinctive n-ary operation f for a *tagh*-transition system specification, we need some notation. Note that the rules of f are totally ordered by the ordering '$\succcurlyeq$'. So, if $m \in nonneg(f)$, there exists a not necessarily unique transition rule ρ, maximal in rank, such that $m \notin pass(\rho)$. In that situation we put $rank(m) = rank(\rho)$ and $act(m) = act(\rho)$, $neg(m) = neg(\rho)$, etc. Also, if, for a 4-tuple R, we have that $R = rank(\rho)$, we put $act(R) = act(\rho)$, $neg(R) = neg(\rho)$, etc. The index set $handle(m)$, the handle of m with respect to f and TSS, is defined as $term(m)$ if $m \in nonneg(f)$, and as $nonneg(f)$ if $m \in neg(f)$.

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

$$handle(m) = \bigcap \{\, term(\rho) \mid m \in act(\rho), \rho \text{ transition rule for } f \,\}.$$

For a negative position m for f, $handle(m)$ simply consists of all non-negative positions. The handles are used in the formulation of the distributive laws below; 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 6 *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:

$$f(\zeta_1, \ldots, z'_p + z''_p, \ldots, \zeta_n) = f(\zeta_1, \ldots, z'_p, \ldots, \zeta_n) + f(\zeta_1, \ldots, z''_p, \ldots, \zeta_n) \quad (4)$$

where $\zeta_q \equiv \varepsilon$ for $q \in \mathrm{handle}(p)$ and $\zeta_q \equiv z_q$ for $q \notin \{p\} \cup \mathrm{handle}(p)$.

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

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

where $\zeta_i \equiv a_i.z'_i$ for $i \in \mathrm{act}(\rho)$, $\zeta_j \equiv \partial^1_{B_j}(z_j)$ for $j \in \mathrm{neg}(\rho)$ with $B_j \neq \emptyset$ and $\zeta_j \equiv z_j$ for $j \in \mathrm{neg}(\rho)$ with $B_j = \emptyset$, $\zeta_k \equiv \varepsilon$ for $k \in \mathrm{term}(\rho)$ and $\zeta_\ell \equiv z_\ell$ for $\ell \in \mathrm{pass}(\rho)$.

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

$$f(\zeta_1, \ldots, \zeta_n) = \delta \quad (6)$$

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

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

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

$$f(\zeta_1, \ldots, \zeta_n) = \varepsilon. \quad (7)$$

where $\zeta_p \equiv \varepsilon$ for $p \in \mathrm{term}(\theta)$ and $\zeta_p \equiv z_p$ for $p \notin \mathrm{term}(\theta)$.

In the distributive laws we demand a 'fingerprint of ε-s' for the particular position instead of allowing a variable for *handle*-arguments. This way, non-determinism at a position is only resolved if it guaranteed that there is termination at sufficiently many other positions, as will be illustrated by the equations in the examples for sequential composition ';' and leftmerge '$\between$' below.

Examples 7 *The transition system specification for ';' generates, according to the definitions above, the following equations:*

$$(x_1 + x_2); y = (x_1; y) + (x_2; y) \qquad (a.x'); y = a.(x'; y) \qquad \varepsilon; \delta = \delta \qquad \varepsilon; \varepsilon = \varepsilon$$
$$\varepsilon; (y_1 + y_2) = (\varepsilon; y_1) + (\varepsilon; y_2) \qquad \varepsilon; (a.y') = a.y' \qquad \delta; y = \delta$$

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 $\delta;(y+b.y')=\delta$, which are special cases of the displayed law $\delta;y=\delta$.

Similarly, we obtain for the leftmerge '$\lfloor\!\lfloor$' the following axiom system:

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

Again we omit the superfluous instantiations of the axiom $\delta;y=\delta$. Note that actually we have exactly the preferred axiomatization, see e.g. [Vra97].

From the termination law $\varepsilon;\varepsilon=\varepsilon$ and $\varepsilon\lfloor\!\lfloor\varepsilon=\varepsilon$ 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 $\varepsilon\lfloor\!\lfloor(a.t+\varepsilon)=\varepsilon$, which is desired for our interpretation of optional termination.

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 transition system specification:

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

$$\frac{y\downarrow}{(x\gg y)\downarrow}$$

The disrupt operator, as can be seen from the transition rules, is a smooth but non-distinctive operation. However, if we split the operation '$\gg$' into two, introducing '$\gg_1$' and '$\gg_2$' say, for which the transition rules satisfy the distinctiveness restrictions, we end up with two smooth and distinctive operations:

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

$$\frac{y\downarrow}{(x\gg_2 y)\downarrow}$$

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 representation of the merge '$\parallel$' in terms

of leftmerge '$\between$', rightmerge '$\between$' and communication merge '$\mid$' using the law $x \parallel y = (x \between y) + (x \between y) + (x \mid y)$.

The same approach, as pointed out in [ABV94] and also applicable for the *tagh*-format, of partitioning of 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.

Definition 8 *Let f be a smooth but non-distinctive n-ary operation for the tagh-transition system specification TSS. The n-ary operations $f_1, \ldots, f_s$ are called distinctive versions of f in a disjoint extension TSS$'$ of TSS if the transition 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*

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

is then referred to as the distinctivity law for f.

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 a smooth operation: there can be a transition rule for f that is not of the format (3), i.e., either there are multiple premises for an action-argument or an active or terminating variable occurs in the target or there is overlap of the index sets or there is no active premise. (Additionally, there can be a position p for which there is no transition rule for f for which this p is non-passive. The latter situation is harmless. See [BV01].)

To illustrate the countermeasure for non-smoothness consider the following, synthesized, one-rule transition system specification adapted from [ABV94]. Here, 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 as active, as terminating and as negative argument).

$$\frac{x \xrightarrow{a} y_1 \quad x \xrightarrow{b} y_2 \quad x \xrightarrow{c} \quad x\!\downarrow}{f(x) \xrightarrow{d} x + y_1} \qquad \frac{x\!\downarrow}{f(x)\!\downarrow}$$

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 specification for which f' is a smooth operation:

$$\frac{x_1 \xrightarrow{a} y_1 \quad x_2 \xrightarrow{b} y_2 \quad x_3 \xrightarrow{c} \quad x_4\!\downarrow}{f'(x_1, x_2, x_3, x_4) \xrightarrow{d} x_3 + y_1} \qquad \frac{x_1\!\downarrow \quad x_2\!\downarrow \quad x_3\!\downarrow \quad x_4\!\downarrow}{f'(x_1, x_2, x_3, x_4)\!\downarrow}$$

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 above ideas for the 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 9 *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 there exist 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', such that*

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

$$\frac{\{ x_i \xrightarrow{a_{ip}} y_{ip} \mid i \in I, p \in P_i \} \quad \{ x_j \overset{b_{jq}}{\nrightarrow} \mid j \in J, q \in Q_j \} \quad \{ x_k{\downarrow} \mid k \in K \}}{f(x_1, \ldots, x_n) \xrightarrow{a} C[x_i, x_j, x_k, x_\ell, y_{ip}]} \tag{9}$$

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

$$\frac{\{ x'_i \xrightarrow{a'_i} y'_i \mid i \in I' \} \quad \{ x'_j \overset{b'_{jq}}{\nrightarrow} \mid j \in J', q \in Q'_j \} \quad \{ x'_k{\downarrow} \mid k \in K' \}}{f'(x'_1, \ldots, x'_m) \xrightarrow{a} C'[x'_j, x'_\ell, y'_{ip}]} \tag{10}$$

such that the mapping $x'_i \xrightarrow{a'_i} y'_i \mapsto x_{\phi(i)} \xrightarrow{a'_i} y_{\psi(i)}$, $x'_j \overset{b'_{jq}}{\nrightarrow} \mapsto x_{\phi(j)} \overset{b'_{jq}}{\nrightarrow}$, $x'_k{\downarrow} \mapsto x_{\phi(k)}{\downarrow}$ 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

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

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.

Example 10 *The 'classical' example of a non-smooth operation is the priority operator θ introduced in [BBK86]. Assuming a partial ordering on '$>$' on Act, the action rules of the unary θ and its binary smoothening θ' are the following:*

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

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

In the above we have described 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 transition system remain unaffected. No rules and/or axioms are removed or added concerning operations already present. Also we have defined smoothening laws (11) and distinctivity laws (8) that connect the original and new operations. For smooth and distinctive operations we have introduced various equations describing distributivity, dynamics, deadlock and termination. Collecting this all together we obtain the notion of the transition system specification and the set of equations *generated* by a *tagh*-transition system specification.

Definition 11 *For a tagh-transition system specification TSS, the tagh-transition system specification TSS' generated by TSS and the equational theory ET' generated by TSS are given by the following procedure:*

Step 0 *Let TSS' disjointly extend TSS and TSS^1_∂. Let ET' contain the equations ET^1_∂ for '$+$' and '∂^1_B' (cf. Definition 1).*

Step 1 *For every non-smooth operation f of TSS not in TSS^1_∂, extend TSS' with the smooth version f' of f and add the smoothening law (11) to ET'.*

Step 2 *For every smooth but non-distinctive operation f of TSS' (as obtained after Step 1) but not in TSS^1_∂, extend TSS' with the distinctive versions $f_1, \ldots, f_s$ and add to ET' the distinctivity law (8).*

Step 3 *For each smooth and distinctive operation f of TSS' (as obtained after Step 2) but not in TSS^1_∂ add to ET' the distributive laws (4), the action laws (5), the deadlock laws (6) and the termination laws (7).*

The discussion above regarding the soundness of the various laws combined with the stratification of the procedure of Definition 11 give rise to the following theorem which is proven in [BV01]. (The transition system specification $TSS_/$ comprises a syntactic representation of the Approximation Induction Principle.)

Theorem 12 *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 equality modulo TSS'.*

Examples 13 *Application of the above procedure yields for the disrupt operator '$\gg$' and the priority operator θ the following generated equational theories:*

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

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

Note that the above axiomatizations are quite natural: The equations for the disrupt operation coincide with those of [BB00]. The axiomatization for the priority operator avoids equations for the auxiliary 'unless' operation '$\lhd$' (cf. [BBK86]).

4 Concluding Remarks

We have introduced the *tagh*-format for structured operational semantics. The *tagh*-format enhances the well-known *GSOS*-format with explicit termination. The format additionally 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*. Examples illustrate the technique and indicate the strength of the approach. The resulting laws are equal or close to the hand-crafted axiomatizations as reported in the literature.

In the technical report [BV01] we prove Theorem 12 which states the soundness and completeness modulo bisimulation of the axiomatization for a transition system specification in *tagh*-format obtained from the procedure of Definition 11. The soundness and completeness result is obtained along the lines set out in [ABV94]; several technical complications arise dealing with termination. The proof of head-normalization in the setting of *tagh* is involved. It requires a detailed case analysis and the notions *handle* and *rank* for non-negative positions. However, grosso modo, the outline of the proofs of soundness and completeness follow the corresponding arguments in [ABV94].

References

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

[Bae00] J.C.M. Baeten. Embedding untimed into timed process algebra: the case for explicit termination. In L. Aceto and B. Victor, editors, *Proc. EXPRESS'00*, pages 45–62. ENTCS 39, 2000.

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

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

[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.

[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.

[BV01] J.C.M. Baeten and E.P. de Vink. Axiomatizing GSOS with termination. Technical Report CSR 01–06, Division of Computer Science, TU/e, 2001. See http://www.win.tue.nl/st/medew/pubbaeten.html.

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

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

[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.