Math. Struct. in Comp. Science (2012), *vol.* 22, *pp.* 291–331. © Cambridge University Press 2012
doi:10.1017/S0960129511000417

Proving the validity of equations in GSOS languages using rule-matching bisimilarity

LUCA ACETO[†‡], MATTEO CIMINI[†] and ANNA INGOLFSDOTTIR[†]

School of Computer Science, Reykjavik University, Menntavegur 1, Nauthólsvík, IS-101 Reykjavík, Iceland
Email: {luca,matteo,annai}@ru.is

Received 26 February 2010; revised 18 October 2010

This paper presents a bisimulation-based method for establishing the soundness of equations between terms constructed using operations whose semantics are specified by rules in the GSOS format of Bloom, Istrail and Meyer. The method is inspired by de Simone's FH-bisimilarity and uses transition rules as schematic transitions in a bisimulation-like relation between open terms. The soundness of the method is proved and examples showing its applicability are provided. The proposed bisimulation-based proof method is incomplete, but we do offer some completeness results for restricted classes of GSOS specifications. An extension of the proof method to the setting of GSOS languages with predicates is also offered.

1. Introduction

Equations play a fundamental role in the development of the theory and practice of process calculi and programming languages since they offer a mathematically appealing and concise way of stating the 'laws of programming' (to borrow the title of a paper by Hoare *et al.* (Hoare *et al.* 1987)) that apply to the language under consideration. In the setting of process calculi, the study of equational axiomatisations of behavioural relations has been a classic area of investigation since, for example, the early work of Hennessy and Milner (Hennessy and Milner 1985; Milner 1984), who offered complete axiom systems for bisimilarity (Park 1981) over the finite and regular fragments of Milner's CCS (Milner 1989). Such axiomatisations capture the essence of bisimilarity over those fragments of CCS in a syntactic and often revealing way, and pave the way for the verification of equivalences between processes by means of theorem proving techniques. Despite these early achievements, the search for axiomatisations of process equivalences that are powerful enough to establish all the valid equations between *open* process terms (that is, terms possibly containing variables) has proved to be a very difficult research

[†] The work of the authors has been partially supported by the projects 'The Equational Logic of Parallel Processes' (nr. 060013021), 'New Developments in Operational Semantics' (nr. 080039021) and 'Meta-theory of Algebraic Process Theories' (nr. 100014021) of the Icelandic Research Fund.
[‡] The paper was revised while Luca Aceto held an Abel Extraordinary Chair at Universidad Complutense de Madrid, Spain, supported by the NILS Mobility Project.

https://doi.org/10.1017/S0960129511000417 Published online by Cambridge University Press

problem; see Aceto *et al.* (2005) for a survey of results in this area. For instance, to the best of our knowledge, there is no known axiomatisation of bisimilarity over recursion-free CCS that is complete over open terms, though stepping stones towards such a result are offered in, for example, Aceto *et al.* (2008; 2009).

The most basic property of any equation is that it be *sound* with respect to the chosen notion of semantics. Soundness proofs are often lengthy, work-intensive and need to be carried out for many equations and languages. It is therefore not surprising that the development of general methods for proving equivalences between open terms in expressive process calculi has received some attention since the early developments of the algebraic theory of processes – see, for example, Bruni *et al.* (2000), Larsen and Liu (1991), Rensink (2000), de Simone (1985) and van Weerdenburg (2008) for some of the work in this area over a period of over 20 years. The current paper offers a contribution to this line of research by developing a bisimulation-based method, which we call *rule-matching bisimilarity*, for establishing the soundness of equations between terms constructed using operations whose semantics are specified by rules in the GSOS format of Bloom, Istrail and Meyer (Bloom *et al.* 1995). Rule-matching bisimilarity is inspired by de Simone's FH-bisimilarity (de Simone 1985) and uses transition rules as transition schemas in a bisimulation-like relation between open terms. We prove that rule-matching bisimilarity is a sound proof method for showing the validity of equations with respect to bisimilarity and provide examples showing its incompleteness.

The incompleteness of rule-matching bisimilarity is not unexpected and raises the question of whether the method is powerful enough to prove the soundness of 'interesting' equations. As a partial answer to this question, we provide examples showing the applicability of our proof method. In particular, our method not only applies to a more expressive rule format than the one proposed in de Simone (1985), but is also a sharpening of de Simone's FH-bisimilarity over de Simone languages. See Section 6, where we apply rule-matching bisimilarity to prove the soundness of the equations in de Simone's 'clock example' – this example was discussed in de Simone (1985) to highlight the incompleteness of FH-bisimilarity. On the theoretical side, we also offer some completeness results for restricted classes of GSOS specifications.

We also extend our main results to the setting of GSOS language specifications with predicates. This extension, though not theoretically deep, is significant from the point of view of applications of rule-matching bisimilarity since the operational semantics of several operations commonly found in the literature on, for example, process algebra is best specified using rules involving the use of predicates as first-class notions.

Overall, we believe that while our conditions are neither necessary nor in general can they be checked algorithmically, they do frequently hold, and they are more accessible to machine support than a direct proof of soundness.

The paper is organised as follows. Sections 2 and 3 introduce the necessary preliminaries on the GSOS rule format that are needed in the remainder of the paper. In particular, Section 3 recalls the notion of a ruloid, which plays a key role in the technical developments to follow. In Section 4, we introduce a simple logic of transition formulae and establish a decidability result for the validity of implications between formulae. Implication between certain kinds of transition formulae that are naturally associated with the premises of

https://doi.org/10.1017/S0960129511000417 Published online by Cambridge University Press

(sets of) ruloids is used in the definition of rule-matching bisimilarity in Section 5. In that section, we prove that rule-matching bisimilarity is a sound method for showing the validity of equations in GSOS languages modulo bisimilarity and provide examples showing its incompleteness. We apply rule-matching bisimilarity to show the validity of some sample equations from the literature on process algebra in Section 6. We then give some partial completeness results for rule-matching bisimilarity in Section 7. Section 8 extends our main results to the setting of GSOS languages with predicates. The paper concludes with a discussion of related and future work in Section 9. For the sake of readability, the proofs of technical results are given separately in a series of appendices.

This paper extends the workshop article Aceto *et al.* (2010a) by providing a more technically detailed description of the background material for this work, proofs of the main technical results and further examples. In addition, Theorem 5.8 and the material in Section 8 are new.

2. Preliminaries

We assume familiarity with the basic notation of process algebra and structural operational semantics – see, for example, Aceto *et al.* (2001), Baeten and Weijland (1990), Bloom *et al.* (1995), Groote and Vaandrager (1992), Hennessy (1988), Hoare (1985), Milner (1989), Mousavi *et al.* (2007) and Plotkin (2004) for more details.

Let Var be a countably infinite set of *process variables* with typical elements x, y. A *signature* Σ consists of a set of *operation symbols* that are disjoint from Var, together with a function *arity* that assigns a natural number to each operation symbol. The set $\mathbb{T}(\Sigma)$ of *terms* built from the operations in Σ and the variables in Var is the least set such that:

— each $x \in \mathsf{Var}$ is a term;

— if f is an operation symbol of arity l, and $P_1, \ldots, P_l$ are terms, then $f(P_1, \ldots, P_l)$ is a term.

We use $P, Q, \ldots$ to range over terms and the symbol $\equiv$ for the relation of syntactic equality on terms. We use $\mathrm{T}(\Sigma)$ to denote the set of *closed* terms over Σ, that is, terms that do not contain variables, and use $p, q, \ldots$ to range over it. An operation symbol f of arity 0 will often be called a *constant* symbol, and the term $f()$ will be abbreviated to f.

In addition to terms, we have *actions*, which are elements of some given non-empty, finite set Act, which is ranged over by a, b, c, d. A *positive transition formula* is a triple of two terms and an action, written $P \xrightarrow{a} P'$. A *negative transition formula* is a pair of a term and an action, written $P \overset{a}{\nrightarrow}$.

A (*closed*) Σ-*substitution* is a function σ from variables to (closed) terms over the signature Σ. For t a term or a transition formula, we write $t\sigma$ for the result of substituting $\sigma(x)$ for each x occurring in t, and $\mathsf{vars}(t)$ for the set of variables occurring in t. A Σ-*context* $C[\vec{x}]$ is a term in which at most the variables $\vec{x}$ appear. $C[\vec{P}]$ is $C[\vec{x}]$ with x_i replaced by P_i wherever it occurs.

https://doi.org/10.1017/S0960129511000417 Published online by Cambridge University Press

Definition 2.1 (GSOS rule). Suppose Σ is a signature. A *GSOS rule* ρ over Σ is a rule of the form

$$\frac{\bigcup_{i=1}^{l} \left\{ x_i \xrightarrow{a_{ij}} y_{ij} \mid 1 \leqslant j \leqslant m_i \right\} \quad \cup \quad \bigcup_{i=1}^{l} \left\{ x_i \overset{b_{ik}}{\nrightarrow} \mid 1 \leqslant k \leqslant n_i \right\}}{f(x_1, \ldots, x_l) \xrightarrow{c} C[\vec{x}, \vec{y}]} \tag{1}$$

where all the variables are distinct, $m_i, n_i \geqslant 0$, a_{ij}, b_{ik} and c are actions, f is an operation symbol from Σ with arity l and $C[\vec{x}, \vec{y}]$ is a Σ-context.

It is useful to name the components of rules:

— The operation symbol f is the *principal operation*.
— The term $f(\vec{x})$ is the *source*.
— $C[\vec{x}, \vec{y}]$ is the *target*.
— c is the *action*.
— The formulae above the line are the *antecedents*, which are sometimes denoted by $\mathsf{ante}(\rho)$.
— The formula below the line is the *consequent*, which is sometimes denoted by $\mathsf{cons}(\rho)$.

For a GSOS rule ρ, we use $\mathsf{SV}(\rho)$ and $\mathsf{TV}(\rho)$ to denote the sets of source and target variables of ρ, respectively: that is, $\mathsf{SV}(\rho)$ is the set of variables in the source of ρ and $\mathsf{TV}(\rho)$ is the set of y's for antecedents $x \xrightarrow{a} y$.

Definition 2.2. A *GSOS language* is a pair $G = (\Sigma_G, R_G)$ where Σ_G is a finite signature and R_G is a finite set of GSOS rules over Σ_G.

Informally, the intent of a GSOS rule is as follows. Suppose we are wondering whether $f(\vec{P})$ is capable of taking a c-step. We look at each rule with principal operation f and action c in turn. We inspect each positive antecedent $x_i \xrightarrow{a_{ij}} y_{ij}$, checking if P_i is capable of taking an a_{ij}-step for each j, and if it is, calling the a_{ij}-children Q_{ij}. We also check the negative antecedents: that is, whether P_i is *in*capable of taking a b_{ik}-step for each k. If these checks succeed, the rule *fires* and $f(\vec{P}) \xrightarrow{c} C[\vec{P}, \vec{Q}]$. This means that the transition relation $\rightarrow_G$ associated with a GSOS language G is the one defined by the rules using structural induction over closed Σ_G-terms.

We will now formally define the transition relation induced by a GSOS language.

Definition 2.3. A *transition relation* over a signature Σ is a relation

$$\rightsquigarrow \subseteq \mathsf{T}(\Sigma) \times \mathsf{Act} \times \mathsf{T}(\Sigma).$$

We write $p \overset{a}{\rightsquigarrow} q$ as an abbreviation for $(p, a, q) \in \rightsquigarrow$.

Definition 2.4. Suppose $\rightsquigarrow$ is a transition relation and σ a closed substitution. For each transition formula φ, the predicate $\rightsquigarrow, \sigma \models \varphi$ is defined by

$$\rightsquigarrow, \sigma \models P \xrightarrow{a} Q \triangleq P\sigma \overset{a}{\rightsquigarrow} Q\sigma$$
$$\rightsquigarrow, \sigma \models P \overset{a}{\nrightarrow} \triangleq \nexists Q : P\sigma \overset{a}{\rightsquigarrow} Q.$$

https://doi.org/10.1017/S0960129511000417 Published online by Cambridge University Press

For H a set of transition formulae, we define

$$\rightsquigarrow, \sigma \models H \quad \stackrel{\Delta}{=} \quad \forall \varphi \in H : \ \rightsquigarrow, \sigma \models \varphi$$

and for

$$\frac{H}{\varphi}$$

a GSOS rule,

$$\rightsquigarrow, \sigma \models \frac{H}{\varphi} \quad \stackrel{\Delta}{=} \quad \left(\rightsquigarrow, \sigma \models H \quad \Longrightarrow \quad \rightsquigarrow, \sigma \models \varphi \right).$$

For t a transition formula, a set of such formulae or a GSOS rule, we sometimes abbreviate $\rightsquigarrow, \sigma \models t$ to $\sigma \models t$ when the transition relation is clear from the context.

Definition 2.5. Suppose G is a GSOS language and $\rightsquigarrow$ is a transition relation over Σ_G. Then $\rightsquigarrow$ is *sound* for G if and only if for every rule $\rho \in R_G$ and every closed Σ_G-substitution σ, we have $\rightsquigarrow, \sigma \models \rho$. A transition $p \stackrel{a}{\rightsquigarrow} q$ is *supported* by some rule

$$\frac{H}{\varphi} \in R_G$$

if and only if there exists a substitution σ such that $\rightsquigarrow, \sigma \models H$ and $\varphi\sigma = (p \stackrel{a}{\rightarrow} q)$. The relation $\rightsquigarrow$ is *supported* by G if and only if each transition in $\rightsquigarrow$ is supported by a rule in R_G.

It is well known that the requirements of soundness and supportedness are sufficient to associate a unique transition relation with each GSOS language.

Lemma 2.6 (Bloom *et al.* 1995). For each GSOS language G there is a unique sound and supported transition relation.

We write $\rightarrow_G$ for the unique sound and supported transition relation for G. We say that a rule ρ is *junk* in G if it does not support any transition of $\rightarrow_G$. For each closed term p, we define

$$\mathit{init}(p) = \{a \in \mathsf{Act} \mid \exists q : \ p \stackrel{a}{\rightarrow}_G q\}.$$

For a GSOS language G, we let

$$\mathit{init}(\mathrm{T}(\Sigma_G)) = \{\mathit{init}(p) \mid p \in \mathrm{T}(\Sigma_G)\}.$$

The basic notion of equivalence among terms of a GSOS language that we will consider in this paper is *bisimulation equivalence* (Milner 1989; Park 1981).

Definition 2.7. Suppose G is a GSOS language. A binary relation $\sim \ \subseteq \mathrm{T}(\Sigma_G) \times \mathrm{T}(\Sigma_G)$ over closed terms is a *bisimulation* if it is symmetric and $p \sim q$ implies, for all $a \in \mathsf{Act}$:

if $p \stackrel{a}{\rightarrow}_G p'$ then, for some q', we have $q \stackrel{a}{\rightarrow}_G q'$ and $p' \sim q'$.

https://doi.org/10.1017/S0960129511000417 Published online by Cambridge University Press

We write $p \leftrightarrow_G q$ if there exists a bisimulation $\sim$ relating p and q. The subscript G will be omitted when it is clear from the context.

It is well known (Bloom *et al.* 1995) that $\leftrightarrow_G$ is a congruence for all operation symbols f of G.

Let $\mathsf{Bisim}(G)$ denote the quotient algebra of closed Σ_G-terms modulo bisimulation. Then, for $P, Q \in \mathbb{T}(\Sigma_G)$,

$$\mathsf{Bisim}(G) \models P = Q \quad \Leftrightarrow \quad (\forall \text{ closed } \Sigma_G\text{-substitutions } \sigma : P\sigma \leftrightarrow_G Q\sigma).$$

In the following, we shall sometimes consider equations that hold over all GSOS languages that extend a GSOS language G with new operation symbols and rules for the new operations. The following notions from Aceto *et al.* (1994) put these extensions on a formal footing.

Definition 2.8. A GSOS language G' is a *disjoint extension* of a GSOS language G if the signature and rules of G' include those of G, and G' introduces no new rules for operations of G.

If G' disjointly extends G, then G' introduces no new outgoing transitions for the closed terms of G. In particular, this means that $P \leftrightarrow_G Q$ if and only if $P \leftrightarrow_{G'} Q$, for $P, Q \in \mathrm{T}(\Sigma_G)$. (More general conservative extension results are discussed in, for example, Fokkink and Verhoef (1998) and Mousavi and Reniers (2005).)

For G a GSOS language, let $\mathsf{BISIM}(G)$ stand for the class of all algebras $\mathsf{Bisim}(G')$ for G' a disjoint extension of G. Thus we have, for $P, Q \in \mathbb{T}(\Sigma_G)$,

$$\mathsf{BISIM}(G) \models P = Q \quad \Leftrightarrow \quad (\forall G' : G' \text{ a disjoint extension of } G \implies \mathsf{Bisim}(G') \models P = Q).$$

Checking the validity of a statement of the form $\mathsf{Bisim}(G) \models P = Q$ or $\mathsf{BISIM}(G) \models P = Q$ according to the above definition is at best very impractical, as it involves establishing bisimilarity of all closed instantiations of the terms P and Q. It would thus be helpful if we had techniques that could be used to do this using only the information obtainable from the terms P and Q themselves. The development of one such technique will be the subject of the remainder of this paper.

2.1. *Eliminating junk rules*

Note that the definition of a GSOS language given above does not exclude *junk* rules, that is, rules that support no transition in $\rightarrow_G$. For example, the rule

$$\frac{x \xrightarrow{a} y, \qquad x \not\xrightarrow{a}}{f(x) \xrightarrow{a} f(y)}$$

https://doi.org/10.1017/S0960129511000417 Published online by Cambridge University Press

has contradictory antecedents and can never fire. It can also be the case that a (seemingly innocuous) rule like

$$\frac{x \xrightarrow{a} y}{f(x) \xrightarrow{b} f(y)}$$

does not support any transition if $\rightarrow_G$ contains no a-transitions. The possible presence of junk rules does not create any problems in the development of the theory of GSOS languages as presented in Aceto *et al.* (1994) and Bloom *et al.* (1995), so the authors of those papers saw no reason to deal with these rules explicitly.

Our aim in the current paper is to develop a test for the validity of equalities between open terms in GSOS languages. The test we shall present in later sections is based on the idea of using GSOS rules as 'abstract transitions' in a bisimulation-like equivalence between open terms. To make this method easier to use, it is thus desirable, though not strictly necessary (see Remark 5.2), to eliminate junk rules from GSOS languages, as these rules would be interpreted as 'potential transitions' from a term, even though they cannot be realised.

Consider, for example, the trivial GSOS language TRIV with unary operations f and g, and rule

$$f(x) \xrightarrow{a} f(x).$$

One can see immediately that $\mathsf{Bisim}(\mathrm{TRIV}) \models f(x) = g(y)$ since the set of closed terms in TRIV is empty. However, if we considered the rule for f as a transition from $f(x)$ in a simple-minded way, we would be led to distinguish $f(x)$ and $g(y)$ because the former has a transition while the latter does not. The rule for f given above is obviously junk.

Clearly, junk rules can be removed from a GSOS language G without altering the associated transition relation. Of course, in order to remove junk rules from a GSOS language, we need to be able to discover effectively which rules are junk. This is indeed possible, as shown by the following theorem, which is due to Aceto, Bloom and Vaandrager – we will reproduce a proof of this result here since we will refer to it in the proofs of Theorems 4.1 and 8.7.

Theorem 2.9 (Aceto *et al.* 2001, Theorem 5.22). We let $G = (\Sigma_G, R_G)$ be a GSOS language and suppose that $\rho \in R_G$. Then, determining whether ρ is junk in G is decidable.

Proof. Let $G = (\Sigma_G, R_G)$ be a GSOS language. We first note that it is easy to determine which rules in R_G are junk once we have computed the set

$$\mathit{init}(\mathrm{T}(\Sigma_G)) \;=\; \{\mathit{init}(p) \mid p \in \mathrm{T}(\Sigma_G)\}$$

where $\mathit{init}(p) = \{a \in \mathsf{Act} \mid \exists q : \; p \xrightarrow{a} q\}$. In fact, one can see immediately that the hypotheses of a GSOS rule of the form (1) are satisfiable if and only if there exist processes $p_1, \ldots, p_l \in \mathrm{T}(\Sigma_G)$ such that

$$\{a_{ij} | 1 \leqslant j \leqslant m_i\} \subseteq \mathit{init}(p_i)$$

https://doi.org/10.1017/S0960129511000417 Published online by Cambridge University Press

and

$$\{b_{ik} | 1 \leqslant k \leqslant n_i\} \cap init(p_i) = \emptyset$$

for all $1 \leqslant i \leqslant l$.

So now we just need to give an effective way of computing the set $init(\mathrm{T}(\Sigma_G))$ for any GSOS language G. This we can do as follows. First note that each function symbol $f \in \Sigma_G$ of arity l determines a computable function

$$\hat{f} : \underbrace{2^{\mathsf{Act}} \times \cdots \times 2^{\mathsf{Act}}}_{l\text{-times}} \to 2^{\mathsf{Act}}$$

by $\hat{f}(X_1, \ldots, X_l) = Y$, where for all $c \in \mathsf{Act}$, we have $c \in Y$ if and only if there exists a rule ρ for f of the form (1) with action c such that for all $1 \leqslant i \leqslant l$, we have $\{a_{ij} | 1 \leqslant j \leqslant m_i\} \subseteq X_i$ and $\{b_{ik} | 1 \leqslant k \leqslant n_i\} \cap X_i = \emptyset$. (If f is a constant, we use $\hat{f}$ to denote the subset Y of Act such that $f(\{\bullet\}) = Y$.) Now, for each $X \subseteq 2^{\mathsf{Act}}$, let $\mathscr{G}(X)$ be given by

$$\mathscr{G}(X) \stackrel{\Delta}{=} \{Y \mid \exists f \in \Sigma_G,\ X_1, \ldots, X_l \in X : \hat{f}(X_1, \ldots, X_l) = Y\}.$$

Note that $\mathscr{G}(X)$ can be effectively computed for each $X \subseteq 2^{\mathsf{Act}}$, and that $X \subseteq Y$ implies $\mathscr{G}(X) \subseteq \mathscr{G}(Y)$.

Our strategy for computing $init(\mathrm{T}(\Sigma_G))$ is to divide the set of closed terms $\mathrm{T}(\Sigma_G)$ into sets T_i of terms with depth less than or equal to i, and to compute the non-decreasing sequence

$$init(T_1) \subseteq init(T_2) \subseteq \cdots$$

until it stabilises. It is obvious that this sequence will stabilise in a finite number of steps since Act is finite.

Now, the set T_1 contains all the constants in the language. Thus $init(T_1)$ can be computed by inspection of the rules for the constants (note that, from the form of these rules, they have no antecedents). In fact, we have

$$init(T_1) = \left\{\hat{f} \mid f \text{ a constant symbol in } \Sigma_G\right\}.$$

So suppose we want to compute $init(T_{i+1})$ given that we already have $init(T_i)$. We claim that $init(T_{i+1}) = \mathscr{G}(init(T_i))$. In fact, each term in T_{i+1} is of the form $f(p_1, \ldots, p_l)$, where the p_i's are all in T_i. Thus we know $init(p_i)$ for all $1 \leqslant i \leqslant l$, and that is exactly what we need to determine which rules for f can fire from that term. Hence, we can compute $init(T_1)$, and each $init(T_{i+1})$ can be computed from $init(T_i)$ using the monotonic and effective operation $\mathscr{G}(\cdot)$. □

Note that a GSOS rule without antecedents, in other words, an axiom, is junk if and only if the set of closed terms is empty. For example, as mentioned before, the rule for the operation f in TRIV is junk.

https://doi.org/10.1017/S0960129511000417 Published online by Cambridge University Press

As a further example, consider the GSOS language G with constant a^ω and unary operation f with rules

$$\frac{}{a^\omega \xrightarrow{a} a^\omega} \qquad \frac{}{f(x) \xrightarrow{a} f(x)} \qquad \frac{x \xrightarrow{b} y}{f(x) \xrightarrow{b} f(y)}.$$

Using our decision procedure, it is easy to check that $\mathit{init}(\mathrm{T}(\Sigma_G)) = \{\{a\}\}$. Thus the rule

$$\frac{x \xrightarrow{b} y}{f(x) \xrightarrow{b} f(y)}$$

is junk in G, as its antecedent cannot be satisfied.

As a consequence of the above theorem, all the junk rules in a GSOS language can be effectively removed in a pre-processing step before applying the techniques described in the subsequent sections, so from now on we will restrict ourselves to GSOS languages without junk rules.

3. Ruloids and the operational specification of contexts

As we mentioned earlier, the essence of our method for checking the validity of equations in GSOS languages is to devise a variation on bisimulation equivalence between contexts that considers GSOS rules as transitions. For primitive operations in a GSOS language G, the rules in R_G will be viewed as abstract transitions from terms of the form $f(\vec{x})$. However, in general, we will be dealing with complex contexts in $\mathbb{T}(\Sigma_G)$. So, in order to apply our ideas to general open terms, we will need to associate with arbitrary contexts a set of derived rules – called *ruloids* (Bloom *et al.* 1995) – describing their behaviour.

A *ruloid* for a context $D[\vec{x}]$, with $\vec{x} = (x_1, \ldots, x_l)$, takes the form

$$\frac{\bigcup_{i=1}^{l} \left\{ x_i \xrightarrow{a_{ij}} y_{ij} \mid 1 \leqslant j \leqslant m_i \right\} \quad \cup \quad \bigcup_{i=1}^{l} \left\{ x_i \overset{b_{ik}}{\nrightarrow} \mid 1 \leqslant k \leqslant n_i \right\}}{D[\vec{x}] \xrightarrow{c} C[\vec{x}, \vec{y}]} \tag{2}$$

where the variables are distinct, $m_i, n_i \geqslant 0$, a_{ij}, b_{ik} and c are actions, and $C[\vec{x}, \vec{y}]$ is a Σ-context.

A ruloid ρ with the above form is sound for $D[\vec{x}]$ if and only if for every closed Σ_G-substitution σ, we have $\rightarrow_G, \sigma \models \rho$. A set of such ruloids is sound for $D[\vec{x}]$ if all of its members are sound.

Definition 3.1. A set of ruloids R is *supporting*[†] for a context $D[\vec{x}]$ and action c if and only if all the consequents of ruloids in R are of the form $D[\vec{x}] \xrightarrow{c} C[\vec{x}, \vec{y}]$ and, whenever $D[\vec{P}] \xrightarrow{c}_G p$, there is a ruloid $\rho \in R$ and a closed substitution σ such that $\mathsf{cons}(\rho)\sigma = D[\vec{P}] \xrightarrow{c} p$ and $\rightarrow_G, \sigma \models \mathsf{ante}(\rho)$.

[†] Our terminology departs slightly from that of Bloom *et al.* (1995) in that we use the term 'supporting' instead of their 'specifically witnessing'.

https://doi.org/10.1017/S0960129511000417 Published online by Cambridge University Press

The following theorem is a slightly sharpened version of the Ruloid Theorem (Bloom *et al.* 1995, Theorem 7.4.3).

Theorem 3.2 (Ruloid theorem). Let G be a GSOS language and $X \subseteq \mathsf{Var}$ be a finite set of variables. For each $D[\vec{x}] \in \mathbb{T}(\Sigma_G)$ and action c, there exists a finite set $R_{D,c}$ of ruloids of the form (2) such that:

(1) $R_{D,c}$ is sound and supporting for $D[\vec{x}]$; and
(2) $\mathsf{TV}(\rho) \cap X = \emptyset$, for every $\rho \in R_{D,c}$.

Moreover, the set $R_{D,c}$ can be effectively constructed.

Proof. The proof is a straightforward adaptation of the proof of the corresponding result in Bloom *et al.* (1995), where we take care in choosing the target variables in ruloids so that Condition (2) in the statement of the theorem is met. □

Definition 3.3. Let G be a GSOS language. For each $D[\vec{x}] \in \mathbb{T}(\Sigma_G)$, the ruloid set of $D[\vec{x}]$, notation $R_G\big(D[\vec{x}]\big)$, is the union of the sets $R_{D,c}$ ($c \in \mathsf{Act}$) given by Theorem 3.2.

Remark 3.4. A set of ruloids that is supporting for a context $D[\vec{x}]$ and action a may have size that is exponential in the number of variables in $\vec{x}$. As an example, consider the context $D[x_1, \ldots, x_n] = f(g(x_1), \ldots, g(x_n))$, where the rules for f and g are as follows.

$$\frac{\left\{x_i \xrightarrow{a} y_i \mid 1 \leqslant i \leqslant n\right\}}{f(x_1, \ldots, x_n) \xrightarrow{a} f(x_1, \ldots, x_n)} \qquad \frac{x \xrightarrow{a} y}{g(x) \xrightarrow{a} y} \qquad \frac{x \xrightarrow{b} y}{g(x) \xrightarrow{a} g(y)}\,.$$

It is easy to see that there are 2^n ruloids for $D[x_1, \ldots, x_n]$ with action a.

The import of the Ruloid Theorem is that the operational semantics of an open term P can be described by a finite set $R_G(P)$ of derived GSOS-like rules. Examples of versions of the above result for more expressive formats of operational rules may be found in, for example, Bloom *et al.* (2004) and Fokkink *et al.* (2006).

Example 3.5. Consider a GSOS language G containing the sequencing operation ';' specified by the following rules (one such pair of rules for each $a \in \mathsf{Act}$):

$$\frac{x \xrightarrow{a} z}{x;y \xrightarrow{a} z;y} \qquad \frac{x \overset{b}{\nrightarrow} \ (\forall b \in \mathsf{Act}), y \xrightarrow{a} z}{x;y \xrightarrow{a} z}\,. \tag{3}$$

Let $R[x,y,z] = x;(y;z)$ and $L[x,y,z] = (x;y);z$. The ruloids for L and R are

$$\begin{array}{ccc} \dfrac{x \xrightarrow{a} x'}{\begin{array}{l} L \xrightarrow{a} (x';y);z \\ R \xrightarrow{a} x';(y;z) \end{array}} & \dfrac{x \nrightarrow, y \xrightarrow{a} y'}{L \xrightarrow{a} y';z \quad R \xrightarrow{a} y';z} & \dfrac{x \nrightarrow, y \nrightarrow, z \xrightarrow{a} z'}{L \xrightarrow{a} z' \quad R \xrightarrow{a} z'} \end{array} \tag{4}$$

where we write $x \nrightarrow$ in the antecedents of ruloids as a shorthand for $x \overset{b}{\nrightarrow} \ (\forall b \in \mathsf{Act})$.

https://doi.org/10.1017/S0960129511000417 Published online by Cambridge University Press

Remark 3.6. Note that the set $R_G\big(D[\vec{x}]\big)$ of ruloids for a context $D[\vec{x}]$ in a GSOS language G may be chosen to contain junk ruloids even when G has no junk rule. For example, consider the GSOS language with constants a and $\mathbf{0}$, unary operation g and binary operation f with the rules

$$\frac{}{a \xrightarrow{a} \mathbf{0}} \qquad \frac{x \xrightarrow{a} x', y \xrightarrow{b} y'}{f(x,y) \xrightarrow{a} \mathbf{0}} \qquad \frac{x \overset{a}{\nrightarrow}}{g(x) \xrightarrow{b} \mathbf{0}}.$$

None of these rules is junk. However, the only ruloid for the context $f(x, g(x))$ is

$$\frac{x \xrightarrow{a} x', x \overset{a}{\nrightarrow}}{f(x,g(x)) \xrightarrow{a} \mathbf{0}},$$

which is junk. However, junk ruloids can be removed from the set of ruloids for a context using Theorem 2.9. In the rest of the paper we shall assume that the set of ruloids under consideration does not contain any junk ruloids.

In the standard theory on GSOS, there is no need to pay much attention to the variables in rules and ruloids, as we are only interested in the transition relation they induce over *closed terms* – in the terminology of Groote and Vaandrager (1992), all the variables occurring in a GSOS rule/ruloid are not *free*. Here, however, we intend to use ruloids as abstract transitions between open terms, and in this framework we need to give a more reasoned account of the role played by variables in ruloids, as the following example shows.

Example 3.7. Consider a GSOS language G containing the unary operations f and g with the rules

$$\frac{x \xrightarrow{a} y}{f(x) \xrightarrow{a} y} \qquad \frac{x \xrightarrow{a} z}{g(x) \xrightarrow{a} z}.$$

It is easy to see that $\mathsf{Bisim}(G) \models f(x) = g(x)$, regardless of the precise description of G. However, in order to prove this equality, any bisimulation-like equivalence relating open terms in $\mathbb{T}(\Sigma_G)$ would have to relate the variables y and z in some way. Of course, this will have to be done carefully since y and z are obviously not equivalent in any non-trivial language.

As this example shows, we will need to develop techniques that allow us to deal with the target variables in ruloids in a reasonable way if we are going to prove many simple equalities between open terms, . In particular, we should not give too much importance to the names of target variables in ruloids.

Definition 3.8 (Valid ruloids). Let G be a GSOS language and $P \in \mathbb{T}(\Sigma_G)$. We say that a ruloid

$$\rho = \frac{H}{P \xrightarrow{a} P'}$$

https://doi.org/10.1017/S0960129511000417 Published online by Cambridge University Press

is *valid* for P if and only if there exist $\rho' \in R_G(P)$ and an injective map $\sigma : \mathsf{TV}(\rho') \to (\mathsf{Var} - \mathsf{SV}(\rho))$ such that ρ is identical to $\rho'\sigma$.

For example, one can see immediately that the rules

$$\frac{x \xrightarrow{a} z}{f(x) \xrightarrow{a} z} \qquad \frac{x \xrightarrow{a} y}{g(x) \xrightarrow{a} y}$$

are valid for the contexts $f(x)$ and $g(x)$ in Example 3.7.

As a further example, consider the unary operation h with rule

$$\frac{x \xrightarrow{a} y_1,\ x \xrightarrow{a} y_2}{h(x) \xrightarrow{a} h(y_1)}\ .$$

Applying the above definition, we immediately have that the rule

$$\frac{x \xrightarrow{a} y_1,\ x \xrightarrow{a} y_2}{h(x) \xrightarrow{a} h(y_2)}$$

is valid for f – just consider a substitution that swaps y_1 with y_2 in the rule for h.

Note, moreover, that each ruloid in $R_G(P)$ is a valid ruloid for P.

The following lemma states that if ρ' is obtained from ρ as in Definition 3.8, then ρ and ρ' are, in some sense, semantically equivalent ruloids.

Lemma 3.9. Let $G = (\Sigma_G, R_G)$ be a GSOS language and $P \in \mathbb{T}(\Sigma_G)$. Assume that ρ is a valid ruloid for P because $\rho = \rho'\sigma$ for some $\rho' \in R_G(P)$ and injective $\sigma : \mathsf{TV}(\rho') \to \mathsf{Var} - \mathsf{SV}(\rho)$. Then:

(1) ρ is sound for $\to_G$.
(2) $\mathsf{Supp}(\rho) = \mathsf{Supp}(\rho')$, where, for a GSOS rule/ruloid $\hat{\rho}$, $\mathsf{Supp}(\hat{\rho})$ denotes the set of transitions supported by $\hat{\rho}$.

The set of valid ruloids for a context P is infinite. However, by Theorem 3.2, we can always select a finite set of valid ruloids for P that is sound and supporting for it. We will often make use of this observation in the rest of the paper.

4. A logic of transition formulae

The set of ruloids associated with an open term P in a GSOS language characterises its behaviour in much the same way as GSOS rules characterise the behaviour of GSOS operations. In fact, by Theorem 3.2, every transition from a closed term of the form $P\sigma$ can be inferred from a ruloid in $R_G(P)$.

The antecedents of ruloids give the precise conditions under which ruloids fire. When matching ruloids in the definition of the bisimulation-like relation between open terms that we aim to define, we will only let a ruloid ρ be matched by a set of ruloids J if the antecedents of ρ are stronger than those of the ruloids in J, that is, if whenever ρ can fire

https://doi.org/10.1017/S0960129511000417 Published online by Cambridge University Press

under a substitution σ, then at least one of the ruloids in J can. In order to formalise this idea, we will make use of a simple propositional logic of initial transition formulae.

We define the language of *initial transition formulae* to be propositional logic with propositions of the form $x \xrightarrow{a}$. Formally, the formulae of such a logic are given by the following grammar:

$$F ::= \quad \mathsf{True} \mid x \xrightarrow{a} \quad \mid \neg F \mid F \wedge F.$$

As usual, we write False for $\neg\mathsf{True}$ and $F \vee F'$ for $\neg(\neg F \wedge \neg F')$.

Let G be a GSOS language. A G-model for initial transition formulae is a substitution σ of processes (closed Σ_G-terms) for variables. We write $\rightarrow_G, \sigma \models F$ if the closed substitution σ is a model of the initial transition formula F. The satisfaction relation $\models$ is defined by structural recursion on F in the obvious way:

$$\begin{array}{lll}
\rightarrow_G, \sigma \models \mathsf{True} & & \text{always} \\
\rightarrow_G, \sigma \models x \xrightarrow{a} & \Leftrightarrow & \sigma(x) \xrightarrow{a}_G p \text{ for some } p \\
\rightarrow_G, \sigma \models \neg F & \Leftrightarrow & \text{not } \rightarrow_G, \sigma \models F \\
\rightarrow_G, \sigma \models F \wedge F' & \Leftrightarrow & \rightarrow_G, \sigma \models F \text{ and } \rightarrow_G, \sigma \models F'.
\end{array}$$

In the following, we consider formulae up to commutativity and associativity of $\vee$ and $\wedge$, and we remove True conjuncts from formulae.

The reader familiar with Hennessy–Milner logic (Hennessy and Milner 1985) will have noticed that the propositions of the form $x \xrightarrow{a}$ correspond to Hennessy–Milner formulae of the form $\langle a \rangle \mathsf{True}$.

If H is a finite set of positive or negative transition formulae (for example, the hypotheses of a rule or ruloid), then $\mathsf{hyps}(H)$ is the conjunction of the corresponding initial transition formulae. Formally,

$$\begin{array}{rcl}
\mathsf{hyps}(\emptyset) & = & \mathsf{True} \\
\mathsf{hyps}\left(\left\{x \overset{a}{\nrightarrow}\right\} \cup H\right) & = & \neg(x \xrightarrow{a}) \wedge \mathsf{hyps}\left(H \setminus \left\{x \overset{a}{\nrightarrow}\right\}\right) \\
\mathsf{hyps}\left(\left\{x \xrightarrow{a} x'\right\} \cup H\right) & = & (x \xrightarrow{a}) \wedge \mathsf{hyps}\left(H \setminus \left\{x \xrightarrow{a} x'\right\}\right).
\end{array}$$

For example,

$$\mathsf{hyps}\left(\left\{x \xrightarrow{a} y, z \overset{b}{\nrightarrow}\right\}\right) = (x \xrightarrow{a}) \wedge \neg(z \xrightarrow{b}).$$

If J is a finite set of ruloids, we overload $\mathsf{hyps}(\cdot)$ and write

$$\mathsf{hyps}(J) \stackrel{\Delta}{=} \bigvee_{\rho' \in J} \mathsf{hyps}\left(\mathsf{ante}(\rho')\right). \tag{5}$$

The semantic entailment preorder between initial transition formulae may now be defined in the standard way: for formulae F, F', we have $\models_G F \Rightarrow F'$ if and only if every substitution that satisfies F must also satisfy F'. Note that a GSOS rule ρ is junk if and only if $\mathsf{hyps}(\rho)$ is semantically equivalent to False.

In the rest of the paper, we will use the semantic entailment preorder between transition formulae in our test for equivalence of open terms to characterise the fact that if one ruloid may fire, then some other may do so too. Of course, in order to do this, we need to

https://doi.org/10.1017/S0960129511000417 Published online by Cambridge University Press

be able to check effectively when $\models_G F \Rightarrow F'$ holds. Fortunately, the semantic entailment preorder between formulae is decidable, as the following theorem shows.

Theorem 4.1. Let G be a GSOS language. Then for all formulae F and F', it is decidable whether $\models_G F \Rightarrow F'$ holds.

Proof sketch. Let F and F' be formulae in the propositional language of initial transition formulae. For each mapping $\eta : \mathsf{vars}(F) \cup \mathsf{vars}(F') \to 2^{\mathsf{Act}}$ and $x \in \mathsf{vars}(F) \cup \mathsf{vars}(F')$, define

$$\eta \models'_G \left(x \xrightarrow{a}\right) \Leftrightarrow a \in \eta(x).$$

We can extend $\models'_G$ to arbitrary formulae with variables in $\mathsf{vars}(F) \cup \mathsf{vars}(F')$ in the obvious way.

It is easy to see that for all $\sigma : \mathsf{Var} \to \mathrm{T}(\Sigma_G)$ and formulae F'' over $\mathsf{vars}(F) \cup \mathsf{vars}(F')$,

$$\to_G, \sigma \models_G F'' \text{ if and only if } \eta_\sigma \models'_G F'',$$

where $\eta_\sigma(x) = \mathit{init}(\sigma(x))$ for all $x \in \mathsf{vars}(F) \cup \mathsf{vars}(F')$.

So, to see that our claim holds, it is now sufficient to note that we can reduce the refinement problem to checking that for every mapping

$$\eta : \mathsf{vars}(F) \cup \mathsf{vars}\left(F'\right) \to \mathit{init}(\mathrm{T}(\Sigma_G)),$$

we have $\eta \models'_G F$ implies $\eta \models'_G F'$. This problem is obviously decidable as there are only finitely many such mappings, and $\mathit{init}(\mathrm{T}(\Sigma_G))$ can be computed following the strategy presented in the proof of Theorem 2.9. □

Theorem 4.1 tells us that we can safely use semantic entailment between formulae in our simple propositional language in the test for the validity of open equations in GSOS languages, which we will present in the next section.

5. Rule-matching bisimulation

We will now give a method for checking the validity of equations in the algebra $\mathsf{Bisim}(G)$ based on a variation of the bisimulation technique. Our approach has strong similarities with, and is a sharpening of, *FH-bisimulation*[†], as proposed in de Simone (1984; 1985).

Definition 5.1 (Rule-matching bisimulation). Let G be a GSOS language. A relation $\approx \subseteq \mathbb{T}(\Sigma_G) \times \mathbb{T}(\Sigma_G)$ is a *rule-matching bisimulation* if it is symmetric and $P \approx Q$ implies that:

For each ruloid

$$\frac{H}{P \xrightarrow{a} P'}$$

[†] FH-bisimilarity checking has been implemented in the tool ECRINS (Doumenc *et al.* 1990; Madelaine and Vergamini 1991).

https://doi.org/10.1017/S0960129511000417 Published online by Cambridge University Press

in the ruloid set of P, there exists a finite set J of valid ruloids for Q such that:

(1) For every

$$\rho' = \frac{H'}{Q \xrightarrow{a'} Q'} \in J,$$

we have:

(a) $a' = a$.

(b) $P' \approx Q'$.

(c) $\big(\mathsf{TV}(\rho') \cup \mathsf{TV}(\rho)\big) \cap \big(\mathsf{SV}(\rho) \cup \mathsf{SV}(\rho')\big) = \emptyset$.

(d) If $y \in \mathsf{TV}(\rho) \cap \mathsf{TV}(\rho')$, then $x \xrightarrow{b} y \in H \cap H'$ for some source variable $x \in \mathsf{SV}(\rho) \cap \mathsf{SV}(\rho')$ and action b.

(2) $\models_G \mathsf{hyps}(\rho) \Rightarrow \mathsf{hyps}(J)$.

We write $P \underline{\leftrightarrow}_G^{RM} Q$ if there exists a rule-matching bisimulation $\approx$ relating P and Q. We sometimes refer to the relation $\underline{\leftrightarrow}_G^{RM}$ as *rule-matching bisimilarity*.

Note that, as the source and target variables of GSOS rules and ruloids are distinct, Condition (1)c is equivalent to $\mathsf{TV}(\rho) \cap \mathsf{SV}(\rho') = \emptyset$ and $\mathsf{TV}(\rho') \cap \mathsf{SV}(\rho) = \emptyset$. Moreover, $\underline{\leftrightarrow}_G^{RM}$ is just standard bisimilarity over closed terms.

Remark 5.2. Note that it is easy to handle junk ruloids when establishing a rule-matching bisimilarity. Indeed, if ρ is a junk ruloid, then $\mathsf{hyps}(\rho)$ is semantically equivalent to False. Therefore, we can use an empty set J of ruloids to 'match ρ' and satisfy requirement (2) in Definition 5.1.

Of course, the notion of rule-matching bisimulation is only reasonable if we can prove that it is sound with respect to the standard extension of bisimulation equivalence to open terms. This is the import of the following theorem.

Theorem 5.3 (Soundness). Let G be a GSOS language. Then, for all $P, Q \in \mathbb{T}(\Sigma_G)$, $P \underline{\leftrightarrow}_G^{RM} Q$ implies $\mathsf{Bisim}(G) \models P = Q$.

Proof. See Appendix A □

This theorem says that when trying to establish the equivalence of two contexts P and Q in a GSOS language G, it is sufficient to exhibit a rule-matching bisimulation relating them. A natural question to ask is whether the notion of rule-matching bisimulation is *complete* with respect to equality in $\mathsf{Bisim}(G)$, in other words, whether $\mathsf{Bisim}(G) \models P = Q$ implies $P \underline{\leftrightarrow}^{RM} Q$, for all $P, Q \in \mathbb{T}(\Sigma_G)$. However, the answer to this question is no, as shown by the following counter-example.

Example 5.4. Consider a GSOS language G consisting of a constant $(a + b)^\omega$ with rules

$$\frac{}{(a+b)^\omega \xrightarrow{a} (a+b)^\omega} \qquad \frac{}{(a+b)^\omega \xrightarrow{b} (a+b)^\omega}$$

https://doi.org/10.1017/S0960129511000417 Published online by Cambridge University Press

and unary function symbols f, g, h and i with rules

$$\frac{x \xrightarrow{a} y_1, x \xrightarrow{b} y_2}{h(x) \xrightarrow{a} f(x)} \qquad \frac{x \xrightarrow{a} y_1, x \xrightarrow{b} y_2}{i(x) \xrightarrow{a} g(x)} \qquad \frac{x \xrightarrow{a} y_1, x \xrightarrow{b} y_2}{f(x) \xrightarrow{a} f(x)} \qquad \frac{x \xrightarrow{a} y_1}{g(x) \xrightarrow{a} g(x)}.$$

First note that no rule in G is junk as the hypotheses of all of these rules are satisfiable.

We claim that $\mathsf{Bisim}(G) \models h(x) = i(x)$. To see this, it is sufficient to note that for all $p \in \mathrm{T}(\Sigma_G)$, we have

$$h(p) \xrightarrow{c} r \;\Leftrightarrow\; p \xrightarrow{a}, \; p \xrightarrow{b}, \; c = a \text{ and } r \equiv f(p)$$
$$i(p) \xrightarrow{c} r \;\Leftrightarrow\; p \xrightarrow{a}, \; p \xrightarrow{b}, \; c = a \text{ and } r \equiv g(p).$$

Moreover, for a term p such that $a, b \in \mathit{init}(p)$, one can see immediately that $f(p) \leftrightarrow g(p)$ since both these terms can only perform action a indefinitely.

However, $h(x)$ and $i(x)$ are *not* rule-matching bisimilar. In fact, in order for $h(x) \leftrightarrow^{RM} i(x)$ to hold, we would need $f(x) \leftrightarrow^{RM} g(x)$, but this does not hold as the unique rule for $g(x)$ cannot be matched by the rule for $f(x)$ because $\not\models (x \xrightarrow{a}) \Rightarrow (x \xrightarrow{a} \wedge\, x \xrightarrow{b})$. Take, for example, a closed substitution σ such that $\sigma(x) \equiv h((a+b)^{\omega})$.

Intuitively, the failure of rule-matching bisimulation in the above example is due to the fact that in order for $\mathsf{Bisim}(G) \models h(x) = i(x)$ to hold, it is sufficient that $f(p)$ and $g(p)$ be bisimilar for those terms p that enable transitions from $h(p)$ and $i(p)$, rather than for arbitrary instantiations.

The above example used GSOS rules containing multiple positive antecedents for the same variable. We will now provide a counter-example to the completeness of rule-matching bisimulation that only uses the format of rules due to de Simone (de Simone 1984; 1985). The point of the example is to show that, in general, one needs some kind of *semantic* information to establish the equivalence of two contexts.

Example 5.5. Consider a GSOS language G consisting of the constants $a, b, \mathbf{0}$ with rules

$$\frac{}{a \xrightarrow{a} \mathbf{0}} \qquad \frac{}{b \xrightarrow{b} \mathbf{0}}$$

and unary function symbols f, g, f' and g' with rules

$$\frac{x \xrightarrow{a} y}{f(x) \xrightarrow{a} f'(y)} \qquad \frac{x \xrightarrow{a} y}{g(x) \xrightarrow{a} g'(y)}$$

$$\frac{x \xrightarrow{b} y}{f'(x) \xrightarrow{b} \mathbf{0}} \qquad \frac{}{f'(x) \xrightarrow{a} \mathbf{0}} \qquad \frac{}{g'(x) \xrightarrow{a} \mathbf{0}}.$$

First note that no rule in G is junk as the hypotheses of all of these rules are satisfiable.

https://doi.org/10.1017/S0960129511000417 Published online by Cambridge University Press

We claim that $\mathsf{Bisim}(G) \models f(x) = g(x)$. To see this, it is sufficient to note that for all $p \in \mathrm{T}(\Sigma_G)$, we have:
(1) $f(p) \xrightarrow{a} f'(p') \Leftrightarrow p \xrightarrow{a} p' \Leftrightarrow g(p) \xrightarrow{a} g'(p')$;
(2) $p \xrightarrow{a} p'$ implies $p' \overset{b}{\nrightarrow}$; and
(3) $p \overset{b}{\nrightarrow}$ implies $f'(p) \leftrightarrow a \leftrightarrow g'(p)$.
However, $f(x)$ and $g(x)$ are *not* rule-matching bisimilar. In fact, in order for $f(x) \underline{\leftrightarrow}^{RM} g(x)$ to hold, we would need $f'(y) \underline{\leftrightarrow}^{RM} g'(y)$, but this is not the case as the rule

$$\frac{y \xrightarrow{b} y'}{f'(y) \xrightarrow{b} \mathbf{0}}$$

for $f'(y)$ cannot be matched by the single axiom for $g'(y)$, as they have a different action.

It is easy to check that Example 5.5 gives an instance of an equation that holds in a language G, but not in all of its disjoint extensions. On the other hand, note that the equation discussed in Example 5.4 is valid in each disjoint extension of the GSOS language considered there.

In the next section we will provide examples that will, hopefully, provide convincing support for the idea that rule-matching bisimulation is a tool that, although it is not complete, can be used to check the validity of many interesting equations.

At this point it is natural to ask whether rule-matching bisimilarity is preserved when taking disjoint extensions, in other words, whether an equation that has been proved to hold in a language G using rule-matching bisimilarity remains sound for each disjoint extension of G. The following example shows that this is not the case.

Example 5.6. Consider a GSOS language G consisting of a constant a^ω with rule $a^\omega \xrightarrow{a} a^\omega$ and unary operations f and g with the rules

$$\frac{x \xrightarrow{a} x'}{f(x) \xrightarrow{a} f(x)} \qquad \frac{y \xrightarrow{a} y'}{g(y) \xrightarrow{a} g(y)}.$$

First note that no rule in G is junk as the hypotheses of both of these rules are satisfiable.

We claim that $\mathsf{Bisim}(G) \models f(x) = g(y)$. To see this, it is sufficient to note that each closed term in the language is bisimilar to a^ω. Moreover, $f(x) \underline{\leftrightarrow}_G^{RM} g(y)$ holds because the formulae $x \xrightarrow{a}$ and $y \xrightarrow{a}$ are logically equivalent in G. On the other hand, consider the disjoint extension G' of G obtained by adding the constant $\mathbf{0}$ with no rules to G. The equivalence $f(x) \underline{\leftrightarrow}_{G'}^{RM} g(y)$ does *not* hold in this disjoint extension because $x \xrightarrow{a}$ does not entail $y \xrightarrow{a}$.

However, rule-matching bisimilarity in language G *is* preserved by taking disjoint extensions if the language G is sufficiently expressive in the sense formalised by the following result – recall that $\mathsf{BISIM}(G)$ denotes the class of all algebras $\mathsf{Bisim}(G')$, for G' a disjoint extension of G.

https://doi.org/10.1017/S0960129511000417 Published online by Cambridge University Press

Theorem 5.7. Let G be a GSOS language such that $\mathit{init}(\mathrm{T}(\Sigma_G)) = 2^{\mathsf{Act}}$. Then, for all $P, Q \in \mathbb{T}(\Sigma_G)$, we have $P \leftrightarrow_G^{RM} Q$ implies $\mathsf{BISIM}(G) \models P = Q$.

Proof. The proof of Theorem 5.3 can be repeated, making use of the observation that for each disjoint extension G' of G, the collection of ruloids in G' for a Σ_G-term P coincides with the collection of ruloids for P in G. Moreover, in light of the proviso of the theorem, $\models_G F \Rightarrow F'$ if and only if $\models_{G'} F \Rightarrow F'$, for all formulae F and F'. □

A conceptually interesting consequence of the above result is that when it is applied to a sufficiently expressive GSOS language G, rule-matching bisimilarity is a proof method that is, in some sense, *monotonic with respect to taking disjoint extensions of the original language*. This means that rule-matching bisimilarity can only prove the validity of equations in G that remain true in all its disjoint extensions. A similar limitation applies to the proof methods presented in, for example, de Simone (1985) and van Weerdenburg (2008).

The condition on the set $\mathit{init}(\mathrm{T}(\Sigma_G))$ in the statement of Theorem 5.7 ensures that each semantic entailment $F \Rightarrow F'$ that holds in the GSOS language G also holds in all its disjoint extensions. We will say that an entailment with the above property is *G-robust*. A rule-matching bisimulation over G is G-robust if all the entailments that need to be checked in item (2) in Definition 5.1 are. We can now formulate the following sharpening of Theorem 5.7.

Theorem 5.8. Let G be a GSOS language. Then:

(1) If we assume that $\approx$ is a G-robust rule-matching bisimulation over G and let G' be a disjoint extension of G, then $\approx$ is a rule-matching bisimulation over G'.
(2) If we let $P, Q \in \mathbb{T}(\Sigma_G)$ and assume that $P \leftrightarrow_G^{RM} Q$ can be shown by establishing a G-robust rule-matching bisimulation over G, then $\mathsf{BISIM}(G) \models P = Q$.

Proof. The first claim can be shown by mimicking the proof of Theorem 5.7, and the second claim follows immediately from the first claim and Theorem 5.3. □

Note that all the examples of rule-matching bisimulations we consider in the following section are robust. Indeed, their robustness can be checked syntactically simply by using the well-known validity of the entailment $F \wedge F' \Rightarrow F$. Therefore, in light of Theorem 5.8, the equivalences we discuss hold in all the disjoint extensions of the language fragments considered.

6. Examples

We shall now present some examples showing applications of the 'rule-matching bisimulation technique'. In particular, we shall show how it can be used to verify some well-known equations found in the literature on process algebra.

https://doi.org/10.1017/S0960129511000417 Published online by Cambridge University Press

6.1. *Commutativity of choice in BCCSP*

Let BCCSP (van Glabbeek 2001; Milner 1989) be the GSOS language containing the constant $\mathbf{0}$, unary prefixing operators $a._-$ ($a \in \mathsf{Act}$) and the binary choice operator $+$ with the following rules:

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

It is well known that the equality $x + y = y + x$ holds in each disjoint extension of BCCSP. It is easy to show this using rule-matching bisimilarity. Indeed, the ruloids for the contexts $x + y$ and $y + x$ are

$$\frac{x \xrightarrow{a} x'}{\begin{array}{l} x + y \xrightarrow{a} x' \\ y + x \xrightarrow{a} x' \end{array}} \qquad \frac{y \xrightarrow{a} y'}{\begin{array}{l} x + y \xrightarrow{a} y' \\ y + x \xrightarrow{a} y' \end{array}}.$$

It is easy to check that the relation

$$\approx \;\triangleq\; \{(x + y, y + x), (y + x, x + y)\} \cup \{(z, z) \mid z \in \mathsf{Var}\}$$

is a rule-matching bisimulation.

6.2. *Associativity of sequencing*

Let G be any GSOS language containing the sequencing operation specified by (3) in Example 3.5. Let $R[x, y, z] = x; (y; z)$ and $L[x, y, z] = (x; y); z$. The ruloids for these two contexts were given in (4).

Consider the symmetric closure of the relation

$$\approx \;\triangleq\; \{(R[x, y, z], L[x, y, z]) \mid x, y, z \in \mathsf{Var}\} \cup \mathscr{I}$$

where $\mathscr{I}$ denotes the identity relation over $\mathbb{T}(\Sigma_G)$. By Theorem 5.3, to show that the contexts L and R are equivalent, it is sufficient to check that what we have just defined is a rule-matching bisimulation. In particular, we need to check the correspondence between the ruloids for these contexts (which is the one given in (4)), and then check that the targets are related by $\approx$ – the verification of these facts is trivial. Thus we have shown that sequencing is associative in any GSOS language that contains the sequencing operation.

The associativity proofs for the standard parallel composition operators found in, for example, ACP, CCS, SCCS and Meije, and for the choice operators in those calculi, for example, the associativity of '+' in BCCSP, follow along similar lines.

6.3. *Commutativity of interleaving parallel composition*

Many standard axiomatisations of behavioural equivalences in the literature, such as those offered in Hennessy and Milner (1985), cannot be used to show by purely equational means

https://doi.org/10.1017/S0960129511000417 Published online by Cambridge University Press

that, for example, parallel composition is commutative and associative. We will now show how this can be easily done using the rule-matching bisimulation technique. We will exemplify the methods by showing that the interleaving parallel composition operation $\|$ (Hoare 1985) is commutative.

The rules for $\|$ are (one pair of rules for each $a \in \mathsf{Act}$)

$$\frac{x \xrightarrow{a} x'}{x \,\|\, y \xrightarrow{a} x' \,\|\, y} \qquad \frac{y \xrightarrow{a} y'}{x \,\|\, y \xrightarrow{a} x \,\|\, y'}. \tag{6}$$

The ruloids for the contexts $x \,\|\, y$ and $y \,\|\, x$ given by Theorem 3.2 are (one pair of ruloids for each $a \in \mathsf{Act}$)

$$\frac{x \xrightarrow{a} x'}{\begin{array}{l} x \,\|\, y \xrightarrow{a} x' \,\|\, y \\ y \,\|\, x \xrightarrow{a} y \,\|\, x' \end{array}} \qquad \frac{y \xrightarrow{a} y'}{\begin{array}{l} x \,\|\, y \xrightarrow{a} x \,\|\, y' \\ y \,\|\, x \xrightarrow{a} y' \,\|\, x \end{array}}.$$

One can now see immediately that the relation $\{(x \,\|\, y, y \,\|\, x) \mid x, y \in \mathsf{Var}\}$ is a rule-matching bisimulation in any GSOS language that includes the interleaving operator. In fact, the correspondence between the ruloids is trivial and the targets are related by the above relation.

6.4. *A distributivity law for sequencing*

We can use the 'rule-matching bisimulation' method to verify the validity of a distributivity axiom for sequencing given in Baeten and Vaandrager (1992).

Let G be a GSOS language that extends BCCSP with the sequencing operation defined by the rules (3). The law whose validity we want to check is

$$(a.x + b.y + y'); z \;=\; a.(x; z) + (b.y + y'); z. \tag{7}$$

Let

$$\begin{aligned} L[x, y, y', z] &\equiv (a.x + b.y + y'); z \\ R[x, y, y', z] &\equiv a.(x; z) + (b.y + y'); z. \end{aligned}$$

We will begin by listing the ruloids for the contexts $L[x, y, y', z]$ and $R[x, y, y', z]$. The ruloids for L and R are

$$\frac{}{\begin{array}{l} L \xrightarrow{a} x; z \\ R \xrightarrow{a} x; z \end{array}} \qquad \frac{}{\begin{array}{l} L \xrightarrow{b} y; z \\ R \xrightarrow{b} y; z \end{array}} \qquad \frac{y' \xrightarrow{c} y''}{\begin{array}{l} L \xrightarrow{c} y''; z \\ R \xrightarrow{c} y''; z \end{array}}$$

where c can be any action in Act. Matching the ruloids for the contexts L and R as explicitly given above, it is easy to show that the relation $\{(L, R), (R, L)\} \cup \mathscr{I}$ is a rule-matching bisimulation in any GSOS language that includes the above-mentioned operators.

https://doi.org/10.1017/S0960129511000417 Published online by Cambridge University Press

We fix a partial, commutative and associative function $\gamma : \mathsf{Act} \times \mathsf{Act} \rightharpoonup \mathsf{Act}$, which describes the synchronisation between actions. The $\|$ operation can be described by the rules (for all $a, b, c \in \mathsf{Act}$)

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

Fig. 1. The rules for $\|$

6.5. *De Simone's clock example*

In his seminal paper de Simone (1985), de Simone presented a bisimulation based technique useful for proving open equations between contexts specified using the so-called de Simone format of operational rules. On page 260 of that paper, de Simone discusses two examples showing that there are valid open equalities between contexts that his technique cannot handle. We shall discuss a variation on one of his examples, the *clock example*, which maintains all the characteristics of the original one in de Simone (1985), and show how rule-matching bisimulations can be used to check the relevant equalities.

Suppose we have a GSOS language G that includes parallel composition with synchronisation $\|$ described by the rules in Figure 1, the interleaving operation $\between$ described by the rules (6), and a constant Ω_{Act} (the *clock over the whole set of actions* in de Simone's terminology) with rules

$$\Omega_{\mathsf{Act}} \xrightarrow{a} \Omega_{\mathsf{Act}} \quad (a \in \mathsf{Act}).$$

Consider the contexts $C[x] \equiv x\|\Omega_{\mathsf{Act}}$ and $D[x] \equiv x \between \Omega_{\mathsf{Act}}$. Regardless of the precise description of G, the terms $C[x]$, $D[x]$ and Ω_{Act} are all equal in $\mathsf{Bisim}(G)$. It is easy to show that this is the case by establishing that the symmetric closures of the relations $\{(C[p], \Omega_{\mathsf{Act}}) \mid p \in \mathrm{T}(\Sigma_G)\}$ and $\{(D[p], \Omega_{\mathsf{Act}}) \mid p \in \mathrm{T}(\Sigma_G)\}$ are bisimulations. However, as de Simone argued in de Simone (1985), his techniques based on FH-bisimilarity cannot be used to establish these equalities . However, we can establish their validity using our rule-matching bisimulation technique as follows.

We begin by computing the ruloids for the contexts $C[x]$ and $D[x]$. These are

$$\frac{}{C[x] \xrightarrow{a} C[x]}\ (a \in \mathsf{Act}) \qquad \frac{x \xrightarrow{a} x'}{C[x] \xrightarrow{a} C[x']}\ (a \in \mathsf{Act}) \qquad \frac{x \xrightarrow{a} x'}{C[x] \xrightarrow{b} C[x']}\ \exists c \in \mathsf{Act} : \gamma(a,c) = b$$

and

$$\frac{}{D[x] \xrightarrow{a} D[x]}\ (a \in \mathsf{Act}) \qquad \frac{x \xrightarrow{a} x'}{D[x] \xrightarrow{a} D[x']}\ (a \in \mathsf{Act}),$$

respectively. It is easy to check that the symmetric closure of the relation

$$\{(C[x], D[z]), (C[x], \Omega_{\mathsf{Act}}), (D[x], \Omega_{\mathsf{Act}}) \mid x, z \in \mathsf{Var}\}$$

https://doi.org/10.1017/S0960129511000417 Published online by Cambridge University Press

is a rule-matching bisimulation. The point is that *any* ruloid for $C[x]$ can be matched by an axiom for $D[z]$, and *vice versa*, in other words, *any* ruloid for $D[z]$ can be matched by an axiom for $C[x]$. This is because it is always the case that $\models (x \xrightarrow{a}) \Rightarrow \mathsf{True}$ for $x \in \mathsf{Var}$.

6.6. *Some equations for while loops*

One of the features of GSOS rules is the fact that they allow the copying of arguments, for example, arguments that are tested positively in a GSOS rule may appear in the target of a rule. Recall that the copying of arguments is not allowed in de Simone's format (de Simone 1984; 1985). An operation whose rules use such a feature is a simple kind of while-loop for concurrent processes.

If we now pick two distinguished actions t and f in Act, we can define a binary operation $\mathsf{while}(_,_)$ that runs its second argument if the first can emit a t action. Formally, the rules for $\mathsf{while}(_,_)$ are (one such rule for each $a \in \mathsf{Act}$)

$$\frac{x \xrightarrow{t} x',\ y \xrightarrow{a} y'}{\mathsf{while}(x,y) \xrightarrow{a} y'; \mathsf{while}(x',y)}$$

where ';' is the sequencing operation described previously.

We will now define a few other operations. An unconditional looping construct can be given by the following rules (one such rule for each $a \in \mathsf{Act}$):

$$\frac{x \xrightarrow{a} x'}{\mathsf{loop}(x) \xrightarrow{a} x'; \mathsf{loop}(x)}\,.$$

Note that the rules for the $\mathsf{loop}(_)$ operation also use copying of arguments.

Let t^ω denote a constant with behaviour given by the rule

$$t^\omega \xrightarrow{t} t^\omega.$$

Let G be any GSOS language that disjointly extends BCCSP with these operations. Then, using the rule-matching bisimulation technique, we can easily prove that the following identities hold in $\mathsf{Bisim}(G)$:

$$\begin{aligned} \mathsf{while}(t^\omega, y) &= \mathsf{loop}(y) \\ \mathsf{while}(f.x, y) &= \mathbf{0} \\ \mathsf{while}(t.x, y) &= y; \mathsf{while}(x, y). \end{aligned}$$

As an example, consider the equation

$$\mathsf{while}(t^\omega, y) = \mathsf{loop}(y)\,.$$

https://doi.org/10.1017/S0960129511000417 Published online by Cambridge University Press

The ruloids for the contexts arising from $\mathsf{while}(t^\omega, y)$ and $\mathsf{loop}(y)$ are listed below – there is one ruloid for each $a \in \mathsf{Act}$:

$$\frac{y \xrightarrow{a} y'}{\mathsf{while}(t^\omega, y) \xrightarrow{a} y'; \mathsf{while}(t^\omega, y)} \qquad \frac{y' \xrightarrow{a} y''}{y'; \mathsf{while}(t^\omega, y) \xrightarrow{a} y''; \mathsf{while}(t^\omega, y)}$$

$$\frac{y' \overset{t}{\not\to},\ y \xrightarrow{a} y''}{y'; \mathsf{while}(t^\omega, y) \xrightarrow{a} y''; \mathsf{while}(t^\omega, y)}$$

$$\frac{y \xrightarrow{a} y'}{\mathsf{loop}(y) \xrightarrow{a} y'; \mathsf{loop}(y)} \qquad \frac{y' \xrightarrow{a} y''}{y'; \mathsf{loop}(y) \xrightarrow{a} y''; \mathsf{loop}(y)} \qquad \frac{y' \overset{t}{\not\to},\ y \xrightarrow{a} y''}{y'; \mathsf{loop}(y) \xrightarrow{a} y''; \mathsf{loop}(y)}.$$

It is easy to check that the symmetric closure of the relation

$$\{(\mathsf{while}(t^\omega, y), \mathsf{loop}(y))\} \cup \{(z; \mathsf{while}(t^\omega, y), z; \mathsf{loop}(y)) \mid z \in \mathsf{Var}\}$$

is a rule-matching bisimulation. Indeed, the above-listed ruloid types with identical premises match one by one.

7. Partial completeness results

We have shown in previous sections that the rule-matching bisimulation technique, though not complete in general, can be used to prove several important equations found in the literature on process algebras. In particular, the soundness of all the equations generated by the methods in Aceto *et al.* (1994) can be proved by exhibiting appropriate rule-matching bisimulations. A natural question to ask is whether there are some classes of contexts for which rule-matching bisimulations give us a complete proof technique for establishing equality between contexts. One such class of contexts is, of course, that of closed terms, as rule-matching bisimilarity coincides with bisimilarity over processes.

We will now present another partial completeness result, this time with respect to a class of contexts we call 'persistent'.

Definition 7.1. Let G be a GSOS language and $P \in \mathbb{T}(\Sigma_G)$. We say that P is *persistent* if and only if each ruloid in $R_G(P)$ is of the form

$$\frac{H}{P \xrightarrow{a} P}$$

for some $a \in \mathsf{Act}$.

Thus, persistent contexts are terms that test their arguments, perform actions according to the results of these tests and then remain unchanged.

Theorem 7.2 (Completeness for persistent contexts). Let G be a GSOS language. Then $\mathsf{Bisim}(G) \models P = Q$ if and only if $P \underline{\leftrightarrow}^{RM} Q$, for all persistent $P, Q \in \mathbb{T}(\Sigma_G)$.

https://doi.org/10.1017/S0960129511000417 Published online by Cambridge University Press

Proof. See Appendix B. □

We will now introduce another class of operations for which rule-matching bisimilarity yields a complete proof method.

Definition 7.3 (Non-inheriting rule). A GSOS rule of the form (1) in Definition 2.1 is *non-inheriting* if none of the variables in $\vec{x}$, namely, the source variables in the rule, occurs in the target of the conclusion of the rule $C[\vec{x}, \vec{y}]$. A GSOS language is *non-inheriting* if each of its rules is. Non-inheriting de Simone rules and languages are defined similarly.

In particular, a non-inheriting rule in de Simone format has the form

$$\frac{\left\{x_i \xrightarrow{a_i} y_i \mid i \in I\right\}}{f(x_1, \ldots, x_n) \xrightarrow{c} t},$$

where $I \subseteq \{1, \ldots, n\}$, all the variables x_i and y_i are distinct, a_i and c are actions, f is an operation symbol from Σ with arity n, and t is a term such that $\mathsf{vars}(t) \subseteq \{y_i \mid i \in I\}$ and each variable occurs in t at most once.

Theorem 7.4. Let G be a non-inheriting GSOS language that for each $P \in \mathbb{T}(\Sigma_G)$ and $c \in \mathsf{Act}$ contains at most one ruloid for P having $c \in \mathsf{Act}$ as action. Let G' be the disjoint extension of G obtained by adding to G the operations and rules of the language BCCSP with Act as set of actions. Let P and Q be terms over Σ_G. Then $\mathsf{Bisim}(G') \models P = Q$ implies $P \underline{\leftrightarrow}_{G'}^{RM} Q$.

Proof. See Appendix C. □

A minor modification of the proof argument in Appendix C yields a partial completeness result for a class of de Simone systems.

Theorem 7.5. Let G be a non-inheriting de Simone language that for each $f \in \Sigma_G$ and $c \in \mathsf{Act}$ contains at most one rule having $f \in \Sigma_G$ as principal operation and $c \in \mathsf{Act}$ as action. Let G' be the disjoint extension of G obtained by adding to G the constant $\mathbf{0}$ and the Act-labelled prefixing operations from the language BCCSP. Let P and Q be terms over Σ_G. Then $\mathsf{Bisim}(G') \models P = Q$ implies $P \underline{\leftrightarrow}_{G'}^{RM} Q$.

For instance, this theorem means that rule-matching bisimilarity can prove all the sound equations between terms constructed using variables and the operations of restriction and injective relabelling from CCS (Milner 1989) and synchronous parallel composition from CSP (Hoare 1985).

8. Extending rule-matching bisimilarity to GSOS with predicates

In this section we extend our main results to the setting of GSOS language specifications with predicates. This extension, though not particularly deep theoretically, is significant from the point of view of applications of rule-matching bisimilarity since the operational semantics of several operations commonly found in the literature on, for example, process algebra is best specified using rules involving the use of predicates as first-class notions.

https://doi.org/10.1017/S0960129511000417 Published online by Cambridge University Press

8.1. *GSOS with predicates*

Given a signature Σ and a set $\mathscr{P}$ of predicate symbols, $Pr\,t$ is a *positive predicate formula* and $\neg Pr\,t$ is a *negative predicate formula*, for each $Pr \in \mathscr{P}$ and $t \in \mathbb{T}(\Sigma)$. The shape of the deduction rules in Definition 2.1 can be adapted to prove predicate formulae too, and to involve them in the premises of rules.

Definition 8.1 (GSOS rule with predicates). Suppose Σ is a signature and $\mathscr{P}$ is a set of predicate symbols. A *GSOS rule with predicates* over Σ and $\mathscr{P}$ is a rule of the form

$$\frac{\bigcup_{i=1}^{l}\left\{x_i \xrightarrow{a_{ij}} y_{ij} \mid 1 \leqslant j \leqslant m_i\right\} \quad \cup \quad \bigcup_{i=1}^{l}\left\{x_i \overset{b_{ik}}{\nrightarrow} \mid 1 \leqslant k \leqslant n_i\right\} \quad \cup \quad PH}{(f(x_1,\ldots,x_l) \xrightarrow{c} C[\vec{x},\vec{y}] \quad \text{or} \quad Pr\,f(x_1,\ldots,x_l))} \tag{8}$$

where $PH = \bigcup_{i=1}^{l}\{Pr_{ik}\,x_i \quad \text{or} \quad \neg Pr_{ik}\,x_i \mid 1 \leqslant k \leqslant o_i\}$, all the variables are distinct, $m_i, n_i, o_i \geqslant 0$, a_{ij}, b_{ik}, and c are actions, f is an operation symbol from Σ with arity l, Pr and Pr_{ik} are predicate symbols and $C[\vec{x},\vec{y}]$ is a Σ-context.

Definition 8.2. A *GSOS language with predicates* is a triple $G = (\Sigma_G, \mathscr{P}_G, R_G)$ where Σ_G is a finite signature, $\mathscr{P}_G$ is a finite set of predicate symbols and R_G is a finite set of GSOS rules over Σ_G and $\mathscr{P}_G$.

The transition relation $\rightarrow_G$ and the set PF_G of provable predicate formulae of the form $Pr\ p$, where p is a closed term, are the ones defined by the rules using structural induction over closed Σ_G-terms, as described in Section 2.

The definition of bisimulation can be extended to a setting with predicates in the standard fashion. In particular, bisimilar terms must satisfy the same predicates.

Example 8.3. As a classic example of an operator whose operational specification involves predicates, consider the standard formulation of the sequential composition operator '$\cdot$'. The following rules make use of the predicate symbol $\downarrow$ – intuitively, the formula $x\downarrow$ means that x successfully terminates:

$$(seq1)\ \frac{x \xrightarrow{a} x'}{x \cdot y \xrightarrow{a} x' \cdot y} \qquad (seq2)\ \frac{x\downarrow \quad y \xrightarrow{a} y'}{x \cdot y \xrightarrow{a} y'} \qquad (seq3)\ \frac{x\downarrow \quad y\downarrow}{(x \cdot y)\downarrow}.$$

Remark 8.4. The reader familiar with Baeten and de Vink (2004) may have noticed that the definition of '$\cdot$' fits the tagh format of Baeten and de Vink. In this format, GSOS languages are extended to involve a single predicate, $\downarrow$, which encodes an explicit notion of successful termination of terms. GSOS languages with predicates generalise this format by considering a finite set of predicates. In contrast to the tagh format, rules may also contain negative predicate formulae in their premises and the rules with predicate formulae as conclusions may have transition formulae in premises. Moreover, since the signature may have more than one predicate symbol, a single argument in every rule can be tested more times in the context of different predicates.

https://doi.org/10.1017/S0960129511000417 Published online by Cambridge University Press

8.2. *A ruloid theorem for GSOS languages with predicates*

A *ruloid* for a context $D[\vec{x}]$, with $\vec{x} = (x_1, \ldots, x_l)$, takes the form

$$\frac{\bigcup_{i=1}^{l} \left\{ x_i \xrightarrow{a_{ij}} y_{ij} \mid 1 \leqslant j \leqslant m_i \right\} \quad \cup \quad \bigcup_{i=1}^{l} \left\{ x_i \overset{b_{ik}}{\nrightarrow} \mid 1 \leqslant k \leqslant n_i \right\} \quad \cup \quad PH}{(D[\vec{x}] \xrightarrow{c} C[\vec{x}, \vec{y}] \quad or \quad Pr\, D[\vec{x}])} \tag{9}$$

where

$$PH = \bigcup_{i=1}^{l} \{ Pr_{ik}\, x_i \quad \text{or} \quad \neg Pr_{ik}\, x_i \mid 1 \leqslant k \leqslant o_i \},$$

all the variables are distinct, $m_i, n_i, o_i \geqslant 0$, a_{ij}, b_{ik}, and c are actions, Pr and Pr_{ik} are predicate symbols and $C[\vec{x}, \vec{y}]$ and $D[\vec{x}]$ are Σ-contexts.

The notion of a supporting set of ruloids for a context $D[\vec{x}]$ and action c, which we introduced in Definition 3.1, can be adapted to ruloids of the aforementioned form in the obvious way. Moreover, in a setting with predicates, we need to consider ruloids that are supporting for a context $D[\vec{x}]$ and a predicate symbol Pr.

Definition 8.5. A set of ruloids R is *supporting* for a context $D[\vec{x}]$ and a predicate symbol Pr if and only if all the consequents of ruloids in R are of the form $Pr\, D[\vec{x}]$ and, whenever $Pr\, D[\vec{P}]$ holds, there is a ruloid $\rho \in R$ and a closed substitution σ such that $\mathsf{cons}(\rho)\sigma = Pr\, D[\vec{P}]$ and $\sigma \models \mathsf{ante}(\rho)$.

The following result can be shown by mimicking the proof of the Ruloid Theorem (Theorem 7.4.3) in Bloom *et al.* (1995).

Theorem 8.6 (Ruloid Theorem). Let G be a GSOS language with predicates. For each $D[\vec{x}] \in \mathbb{T}(\Sigma_G)$ and action c, there exists a finite set $R_{D,c}$ of ruloids of the form (9) that is sound and supporting for $D[\vec{x}]$ and action c. Similarly, for each $D[\vec{x}] \in \mathbb{T}(\Sigma_G)$ and predicate symbol Pr, there exists a finite set $R_{D,Pr}$ of ruloids of the form (9) that is sound and supporting for $D[\vec{x}]$ and Pr.

The notion of a valid ruloid given in Definition 3.8 and related results can be adapted naturally to a setting with predicates.

8.3. *The logic of initial transitions with predicates*

In this section we extend our discussion to the logic of initial transition with predicates. Formulae are given by adding propositions of the form $Pr\, x$ to the grammar from Section 4:

$$F ::= \ldots \mid Pr\, x.$$

The semantics of formulae is extended to consider the new kind of predicate formulae and is presented below – we write $(\rightarrow_G \cup PF_G), \sigma \models F$ if the closed substitution σ is a model of the initial transition formula F, where PF_G denotes the set of provable predicate

https://doi.org/10.1017/S0960129511000417 Published online by Cambridge University Press

formulae in G:

$$\begin{array}{lll}
(\rightarrow_G \cup PF_G), \sigma \models \mathsf{True} & & \text{always} \\
(\rightarrow_G \cup PF_G), \sigma \models Pr\, x & \Leftrightarrow & Pr\, \sigma(x) \in PF_G \\
(\rightarrow_G \cup PF_G), \sigma \models x \xrightarrow{a} & \Leftrightarrow & \sigma(x) \xrightarrow{a}_G \\
(\rightarrow_G \cup PF_G), \sigma \models \neg F & \Leftrightarrow & \text{not } (\rightarrow_G \cup PF_G), \sigma \models F \\
(\rightarrow_G \cup PF_G), \sigma \models F \wedge F' & \Leftrightarrow & (\rightarrow_G \cup PF_G), \sigma \models F \text{ and } (\rightarrow_G \cup PF_G), \sigma \models F'.
\end{array}$$

If H is a set of transition or predicate formulae (for example, the hypotheses of a rule or ruloid with predicates), then $\mathsf{hyps}(H)$ is the conjunction of the corresponding initial transition formulae. Formally, the definition in Section 4 is extended by adding the clauses

$$\begin{aligned}
\mathsf{hyps}\left(\{Pr\, x\} \cup H\right) &= (Pr\, x) \wedge \mathsf{hyps}\left(H \setminus \{Pr\, x\}\right) \\
\mathsf{hyps}\left(\{\neg Pr\, x\} \cup H\right) &= \neg(Pr\, x) \wedge \mathsf{hyps}\left(H \setminus \{\neg Pr\, x\}\right).
\end{aligned}$$

The notion of entailment between formulae in the logic with predicate formulae is defined as before.

Theorem 8.7. Let G be a GSOS language with predicates. Then for all formulae F and F', determining whether $\models_G F \Rightarrow F'$ holds is decidable.

Proof sketch. Let F and F' be formulae in the propositional language of initial transition formulae with predicates. For each mapping $\eta : \mathsf{vars}(F) \cup \mathsf{vars}(F') \rightarrow 2^{\mathsf{Act} \cup \mathscr{P}}$ and $x \in \mathsf{vars}(F) \cup \mathsf{vars}(F')$, we define

$$\begin{aligned}
\eta \models'_G (x \xrightarrow{a}) &\Leftrightarrow a \in \eta(x) \\
\eta \models'_G Pr\, x &\Leftrightarrow Pr \in \eta(x).
\end{aligned}$$

We extend $\models'_G$ to arbitrary formulae with variables in $\mathsf{vars}(F) \cup \mathsf{vars}(F')$ in the obvious way.

Now consider the set $init{+}pred(p)$, with $p \in \mathrm{T}(\Sigma_G)$, containing the labels of the initial transitions of the closed term p together with the names of the predicate symbols that p satisfies. Formally, for each closed term p, we define

$$init{+}pred(p) = init(p) \cup \{Pr \in \mathscr{P} \mid Pr\, p\}.$$

It is easy to see that for all $\sigma : \mathsf{Var} \rightarrow \mathrm{T}(\Sigma_G)$ and formulae F'' over $\mathsf{vars}(F) \cup \mathsf{vars}(F')$,

$$(\rightarrow_G \cup PF_G), \sigma \models_G F'' \text{ if and only if } \eta_\sigma \models'_G F'',$$

where $\eta_\sigma(x) = init{+}pred(\sigma(x))$ for all $x \in \mathsf{vars}(F) \cup \mathsf{vars}(F')$.

To see that the claim holds, it is now sufficient to note that we can reduce the refinement problem in this way to the problem of checking that for every mapping

$$\eta : \mathsf{vars}(F) \cup \mathsf{vars}(F') \rightarrow init{+}pred(\mathrm{T}(\Sigma_G)),$$

if $\eta \models'_G F$, then $\eta \models'_G F'$. This problem is obviously decidable as there are only finitely many such mappings, and $init{+}pred(\mathrm{T}(\Sigma_G))$ can be effectively computed using a variant of the strategy used in the proof of Theorem 2.9. □

https://doi.org/10.1017/S0960129511000417 Published online by Cambridge University Press

8.4. *Rule-matching bisimilarity*

In this section we reformulate the notion of rule-matching bisimilarity for GSOS languages with predicates. In the following definition, it may be helpful to bear in mind that in this setting the ruloids for a context P take one of the two forms

$$\frac{H}{P \xrightarrow{a} P'} \quad \text{or} \quad \frac{H}{P r\, P},$$

while in GSOS languages, ruloids can only take the first of these forms. Another important difference is that the set of premises H can involve predicates in GSOS languages with predicates.

Definition 8.8 (Rule-matching bisimulation, reprise). Let G be a GSOS language with predicates. A relation $\approx \subseteq \mathbb{T}(\Sigma_G) \times \mathbb{T}(\Sigma_G)$ is a *rule-matching bisimulation* if it is symmetric and $P \approx Q$ implies that for each ruloid ρ in the ruloid set of P, the following conditions are met.

(1) If ρ is of the form

$$\frac{H}{P \xrightarrow{a} P'},$$

then there exists a finite set J of valid ruloids for Q such that:

— For every

$$\rho' = \frac{H'}{Q \xrightarrow{a'} Q'} \in J,$$

we have:

- $a' = a$,
- $P' \approx Q'$,
- $\big(\mathsf{TV}(\rho') \cup \mathsf{TV}(\rho)\big) \cap \big(\mathsf{SV}(\rho) \cup \mathsf{SV}(\rho')\big) = \emptyset$ and
- if $y \in \mathsf{TV}(\rho) \cap \mathsf{TV}(\rho')$, then $x \xrightarrow{b} y \in H \cap H'$ for some source variable $x \in \mathsf{SV}(\rho) \cap \mathsf{SV}(\rho')$ and action b.

— $\models_G \mathsf{hyps}(\rho) \Rightarrow \mathsf{hyps}(J)$.

(2) If ρ is of the form

$$\frac{H}{P r\, P},$$

then there exists a finite set J of valid ruloids for Q with conclusion $P r\, Q$ such that:

— For every

$$\rho' = \frac{H'}{P r\, Q} \in J,$$

we have:

https://doi.org/10.1017/S0960129511000417 Published online by Cambridge University Press

- $\big(\mathsf{TV}(\rho') \cup \mathsf{TV}(\rho)\big) \cap \big(\mathsf{SV}(\rho) \cup \mathsf{SV}(\rho')\big) = \varnothing$.
- If $y \in \mathsf{TV}(\rho) \cap \mathsf{TV}(\rho')$, then $x \xrightarrow{b} y \in H \cap H'$ for some source variable $x \in \mathsf{SV}(\rho) \cap \mathsf{SV}(\rho')$ and action b.

— $\models_G \mathsf{hyps}(\rho) \Rightarrow \mathsf{hyps}(J)$.

We write $P \underline{\leftrightarrow}_G^{RMp} Q$ if there exists a rule-matching bisimulation $\approx$ relating P and Q. As before, we refer to the relation $\underline{\leftrightarrow}_G^{RMp}$ as *rule-matching bisimilarity*.

Theorem 8.9 (Conservative extension). If G is a GSOS language, then $P \underline{\leftrightarrow}_G^{RM} Q$ if and only if $P \underline{\leftrightarrow}_G^{RMp} Q$ for all $P, Q \in \mathbb{T}(\Sigma_G)$.

Proof. Since G is a GSOS language, every ruloid for any term P has the form

$$\frac{H}{P \xrightarrow{a} P'},$$

where the premises in H do not use predicates. So $\mathsf{hyps}(H)$ does not involve predicate formulae either. So clause (1) of Definition 8.8 is satisfied, and this exactly matches with Definition 5.1 of $\underline{\leftrightarrow}_G^{RM}$. □

It follows from Theorem 8.9 that rule-matching bisimilarity is incomplete over GSOS languages with predicates, since it is not even complete within the framework of standard GSOS – Examples 5.4 and 5.5 also witness incompleteness for $\underline{\leftrightarrow}_G^{RMp}$.

Theorem 8.10 (Soundness). If G is a GSOS language with predicates, then for all $P, Q \in \mathbb{T}(\Sigma_G)$, we have $P \underline{\leftrightarrow}_G^{RMp} Q$ implies $\mathsf{Bisim}(G) \models P = Q$.

Proof. The proof follows along the same lines as the proof of Theorem 5.3. To show that $P \underline{\leftrightarrow}_G^{RMp} Q$ implies $\mathsf{Bisim}(G) \models P = Q$, it is sufficient to prove that the relation $\sim$ given by

$$\sim = \{(P\sigma, Q\sigma) \mid P \underline{\leftrightarrow}_G^{RMp} Q,\ \sigma \text{ a closed substitution}\}$$

is a bisimulation. The addition of predicates adds no complications to the rest of the proof, which is therefore omitted here. □

Following along the lines of the proof of Theorem 5.8, we obtain the following result – the notion of disjoint extension used in the statement is the obvious modification of the one over GSOS languages (see Definition 2.8) for a setting with predicates: in particular, if G' disjointly extends G, then G' adds no new rules for the predicate symbols in G.

Theorem 8.11. Let G be a GSOS language with predicates.

(1) Assume that $\approx$ is a G-robust rule-matching bisimulation over G, and let G' be a disjoint extension of G. Then $\approx$ is a rule-matching bisimulation over G'.
(2) Let $P, Q \in \mathbb{T}(\Sigma_G)$ and assume that $P \underline{\leftrightarrow}_G^{RMp} Q$ can be shown by establishing a G-robust rule-matching bisimulation over G. Then $\mathsf{BISIM}(G) \models P = Q$.

https://doi.org/10.1017/S0960129511000417 Published online by Cambridge University Press

All of the examples of rule-matching bisimulation in a setting with predicates that we have met so far in our analysis of sound equations taken from the literature on process algebra are robust.

8.5. *Examples*

We now present some examples of the application of rule-matching bisimilarity to a setting with predicates.

Example 8.12 (Sequential composition: zero element). Consider the sequential composition operator from Example 8.3. Recall that the constant a^ω is defined by the single axiom $a^\omega \xrightarrow{a} a^\omega$. This constant simply displays a infinitely many times. a^ω is a left-zero element for '$\cdot$' because its infinite behaviour is enough to preempt the execution of the right-hand argument of '$\cdot$'. In this example we will check the law $a^\omega \cdot y \leftrightarrow a^\omega$ by means of rule-matching bisimilarity. To this end, it is sufficient to show that the relation $\{(a^\omega \cdot y, a^\omega), (a^\omega, a^\omega \cdot y)\}$ is a rule-matching bisimilarity.

In constructing the set of ruloids for $a^\omega \cdot y$, rules (seq2) and (seq3) in Example 8.3 play no role because the premise $x\downarrow$ is not satisfied when x is instantiated with the constant a^ω. The only ruloid generated by the ruloid theorem comes from the instantiation of rule (seq1) and is the axiom $a^\omega \cdot y \xrightarrow{a} a^\omega \cdot y$. The set of ruloids for the constant a^ω consists of the singleton set containing the axiom defining the constant. Since the two ruloids match, meeting all the constraints of Definition 8.8, we can use the soundness theorem (Theorem 8.10) to conclude that a^ω is a left zero element for $\cdot$.

Example 8.13 (Sequential composition: associativity). In this example, we will prove that the sequential composition operator '$\cdot$' considered in the previous example is associative.

We let $R[x,y,z] = x\cdot(y\cdot z)$ and $L[x,y,z] = (x\cdot y)\cdot z$, and consider the symmetric closure of the relation

$$\approx \stackrel{\Delta}{=} \{(R[x,y,z], L[x,y,z]) \mid x,y,z \in \mathsf{Var}\} \cup \mathscr{I}$$

where $\mathscr{I}$ denotes the identity relation over $\mathbb{T}(\Sigma_G)$.

As in the case of associativity of the sequencing operator considered in Section 6, for clarity, we can present the ruloids of R and L and their 'matching' as follows:

$$\frac{x \xrightarrow{a} x'}{\begin{array}{l} L \xrightarrow{a} (x'\cdot y)\cdot z \\ R \xrightarrow{a} x'\cdot(y\cdot z) \end{array}} \qquad \frac{x\downarrow, \quad y \xrightarrow{a} y'}{\begin{array}{l} L \xrightarrow{a} y'\cdot z \\ R \xrightarrow{a} y'\cdot z \end{array}} \qquad \frac{x\downarrow, \quad y\downarrow, \quad z \xrightarrow{a} z'}{\begin{array}{l} L \xrightarrow{a} z' \\ R \xrightarrow{a} z' \end{array}} \qquad \frac{x\downarrow, \quad y\downarrow, \quad z\downarrow}{\begin{array}{l} L\downarrow \\ R\downarrow \end{array}}.$$

It is now easy to see that any ruloid for R matches with its corresponding one for L satisfying the conditions of Definition 8.8, and *vice versa*. The relation $\approx$ is indeed a rule-matching bisimulation and, by the soundness theorem, the associativity of this sequential composition operator then follows.

Example 8.14 (Predictable failure constant of $BPA_{0\delta}$). In this example we focus on Baeten and Bergstra's $BPA_{0\delta}$ (Baeten and Bergstra 1990). The *predictable failure* 0 is a constant

https://doi.org/10.1017/S0960129511000417 Published online by Cambridge University Press

that absorbs the computation no matter where it appears within the context of the (non-standard) sequential composition operator '$\odot$'[†], in other words, 0 is a zero element for the operator '$\odot$'. The laws $x \odot 0 \leftrightarrow 0 \odot x \leftrightarrow 0$ both hold, and the purpose of this example is to check them using $\underline{\leftrightarrow}^{RMp}$. The following SOS rules make use of the predicate $\neq 0$ that determines whether or not a process can be proved equal to 0, and of predicates $\xrightarrow{a} \checkmark$ that tell us when a process can terminate by performing an a action:

$$\frac{x \neq 0 \quad y \neq 0}{(x \odot y) \neq 0} \qquad \frac{x \xrightarrow{a} x' \quad y \neq 0}{x \odot y \xrightarrow{a} x' \odot y} \qquad \frac{x \xrightarrow{a} \checkmark \quad y \neq 0}{x \odot y \xrightarrow{a} y}.$$

Consider the equation $0 \odot x \leftrightarrow 0$. Let $L[x] = 0 \odot x$ and $R = 0$. The set of ruloids for R is clearly empty, since 0 is defined by no rules. Also, the set of ruloids for L is empty since the premises $x \neq 0$, $x \xrightarrow{a} x'$ and $x \xrightarrow{a} \checkmark$ are not satisfied when x is instantiated to 0. The contexts L and R are thus trivially equated by $\underline{\leftrightarrow}^{RMp}$. It is easy to see that the law $x \odot 0 \leftrightarrow 0$ can be checked in a similar way. The ruloid set for $x \odot 0$ is again empty since the premise $y \neq 0$, contained in all of the three rules above, is not satisfied when y is instantiated to 0. Thanks to Theorem 8.10, this is sufficient for us to conclude that the constant 0 is both a left and right zero element for '$\odot$'.

9. Related and future work

The development of general methods for proving equivalences between open terms in expressive process calculi is a challenging subject, which has received attention since the early developments of the algebraic theory of processes – see, for example, Bruni *et al.* (2000), Larsen and Liu (1991), Rensink (2000), de Simone (1985) and van Weerdenburg (2008) for some of the work in this area. De Simone's FH-bisimilarity (de Simone 1985) represents a meaningful early step towards a general account of the problem by presenting, for the first time, a sound bisimulation method in place of the usual definition, which involves the closure under all possible substitutions. Our method relies mainly on the concepts underlying FH-bisimilarity and is a refinement of that notion in the more expressive setting of GSOS languages – see de Simone's 'Clock Example' discussed in Section 6.5, where FH-bisimilarity fails but $\underline{\leftrightarrow}^{RM}$ succeeds.

Later, Rensink (2000) addressed the problem of checking bisimilarity of open terms using a natural sharpening of de Simone's FH-bisimilarity. Rensink's extension of FH-bisimilarity is orthogonal to ours, and provides another method for checking equivalences between open terms that is more powerful than the original FH-bisimilarity. Rensink defined a new notion of bisimulation equivalence, called *hypothesis preserving bisimilarity*, that adds to FH-bisimilarity the capability of storing some kinds of information about the variable transitions during the computation.

To explain the import of hypothesis preserving bisimilarity, we can look at Example 5.4. First note that $\underline{\leftrightarrow}^{RM}$ fails to establish the sound equation $h(x) = i(x)$ because at the

[†] Baeten and Bergstra use $\cdot$ to denote the sequential composition operator in $BPA_{0\delta}$, but we use $\odot$ here to avoid any confusion with the standard sequential composition operator considered in Example 8.3.

https://doi.org/10.1017/S0960129511000417 Published online by Cambridge University Press

second step of the computation some knowledge about the transitions of the closed term p substituted for x has already been established (indeed, we know at that point that p performs a b-transition since this was tested for at the first step). However, when comparing $f(x)$ and $g(x)$, rule-matching bisimulation behaves in a memoryless fashion and ignores this information. Rensink's hypothesis preserving bisimilarity takes into account the history, and this is enough to overcome the difficulties in that example and analogous scenarios. Adding this feature to $\underline{\leftrightarrow}^{RM}$ would lead to a more powerful rule-matching equivalence; we leave this further sharpening for future work, together with extensions of $\underline{\leftrightarrow}^{RM}$ to more expressive rule formats.

Recently, van Weerdenburg (2008) addressed the automation of soundness proofs. His approach differs from ours and that in Rensink (2000) and de Simone (1985), since he translates the operational semantics into a logical framework. In such a framework, rules are encoded as logical formulae and the overall semantics turns out to be a logical theory, for which van Weerdenburg provides a sequent calculus style proof system. In order to highlight the applicability of his method, Van Weerdenburg (2008) provides some examples of equivalences from the literature that can be proved using it. However, although the ultimate aim of the research described in van Weerdenburg (2008) is the automation of soundness proofs, his system does have some drawbacks. The main one being that the user is not only required to provide both the operational semantics and the equation to check (together with the standard encoding of bisimilarity), but must also provide a candidate bisimulation relation that can be used to show the validity of the equation under consideration, together with all the axioms that are needed to complete the proof. This means the user must have a clear understanding of what the proof is going to look like, which seems to be a general and inescapable drawback when approaching the problem of checking equations through a translation into a logical system.

Despite this minor drawback, the approach proposed by van Weerdenburg is very interesting and complements the proposals based on the ideas underlying de Simone's FH-bisimilarity, including ours. We believe that an adequate solution to the problem of automating checks for the validity of equations in process calculi will be based on a combination of bisimulation-based and logical approaches.

A related line of work has been pursued in, for example, Aceto *et al.* (2010), Aceto *et al.* (2010b), Cranen *et al.* (2008) and Mousavi *et al.* (2005), which present rule formats that guarantee the soundness of certain algebraic laws over a process language 'by design', provided the SOS rules giving the semantics of certain operators fit that format. This is an orthogonal line of investigation to the one reported in the current paper. As a test case for the applicability of our rule-based bisimilarity, we have checked that the soundness of all the equations guaranteed to hold by the commutativity format given in Mousavi *et al.* (2005) can be shown using $\underline{\leftrightarrow}^{RM}$, and we are now carrying out similar investigations for the rule formats proposed in Aceto *et al.* (2010) and Cranen *et al.* (2008).

Another avenue of future research that we are actively pursuing is the search for more, and more general, examples of partial completeness results for rule-matching bisimulation over GSOS and de Simone languages. Indeed, the partial completeness results we present in Section 7 are just preliminary steps and leave substantial room for improvement. Last,

https://doi.org/10.1017/S0960129511000417 Published online by Cambridge University Press

but not least, we are about to start work on the implementation of a prototype checker for rule-matching bisimilarity.

Appendix A. Proof of Theorem 5.3

Definition A.1. Let f and g be functions, and S a subset of both their domains. Then $f = g$ on S means $\forall x \in S. f(x) = g(x)$.

The following basic lemma about $\models$ will be useful in the proof.

Lemma A.2. We let σ, σ' be closed substitutions and $S \subseteq \mathsf{Var}$ and assume that $\sigma = \sigma'$ on S. Then, for all initial transition formulae F such that $\mathsf{vars}(F) \subseteq S$,

$$\rightarrow_G, \sigma \models F \Leftrightarrow \rightarrow_G, \sigma' \models F.$$

We are now ready to embark on the proof of Theorem 5.3.

Proof of Theorem 5.3. To show that $P \underline{\leftrightarrow}_G^{RM} Q$ implies $\mathsf{Bisim}(G) \models P = Q$, it is sufficient to prove that the relation $\sim$ given by

$$\sim = \{(P\sigma, Q\sigma) \mid P \underline{\leftrightarrow}_G^{RM} Q,\ \sigma \text{ a closed substitution}\}$$

is a bisimulation. We first note that $\sim$ is symmetric since $\underline{\leftrightarrow}_G^{RM}$ is.

So we assume that $P\sigma \sim Q\sigma$, and that $P\sigma \xrightarrow{a} p$. We will show that $Q\sigma \xrightarrow{a} q$ for some q such that $p \sim q$. By the Ruloid Theorem, $P\sigma \xrightarrow{a} p$ because there is some ruloid

$$\rho = \frac{H}{P \xrightarrow{a} P'} \in R_G(P)$$

enabling this transition, and some substitution σ' such that:

(1) $\sigma' \models H$,
(2) $P\sigma' \equiv P\sigma$, and
(3) $P'\sigma' \equiv p$.

Note that, by (1), we have $\sigma' \models \mathsf{hyps}(\rho)$, and, by (2), $\sigma = \sigma'$ on $\mathsf{vars}(P)$. Thus, as the variables in $\mathsf{hyps}(\rho)$ are included in $\mathsf{vars}(P)$, Lemma A.2 gives us $\sigma \models \mathsf{hyps}(\rho)$.

As $P \underline{\leftrightarrow}^{RM} Q$, there exists a set J of valid ruloids for Q satisfying the conditions in Definition 5.1. In particular, by the condition Definition 5.1 (1) c, no target variable in a ruloid $\rho' \in J$ occurs as a source variable in ρ and, conversely, no source variable in a ruloid $\rho' \in J$ occurs as a target variable in ρ.

Our aim now is to find a move from $Q\sigma$ matching the transition $P\sigma \equiv P\sigma' \xrightarrow{a} P'\sigma' \equiv p$. To this end, we will construct a substitution τ and find a ruloid

$$\rho' = \frac{H'}{Q \xrightarrow{a} Q'} \in J$$

such that

$$\sigma = \tau \text{ on } \mathsf{vars}(P) \cup \mathsf{vars}(Q) \tag{10}$$

$$\tau \models H' \tag{11}$$

$$\sigma' = \tau \text{ on } \mathsf{TV}(\rho). \tag{12}$$

https://doi.org/10.1017/S0960129511000417 Published online by Cambridge University Press

Note that (10) implies that $P\sigma \equiv P\sigma' \equiv P\tau$ and $Q\sigma \equiv Q\tau$. Moreover, (10) and (12) imply $\sigma' = \tau$ on $\mathsf{vars}(\rho)$, and thus that $p \equiv P'\sigma' \equiv P'\tau$. Condition (11) will make sure that the selected ruloid fires under the substitution τ.

To this end, consider the substitution σ'' given by

$$\sigma''(x) = \begin{cases} \sigma(x) & x \in \mathsf{vars}(P) \cup \mathsf{vars}(Q) \\ \sigma'(x) & \text{otherwise.} \end{cases}$$

Note that, as $\mathsf{TV}(\rho) \cap \mathsf{vars}(Q) = \emptyset$, $\sigma'' = \sigma'$ on $\mathsf{vars}(\rho)$. So $\sigma'' \models H$ and, *a fortiori*, $\sigma'' \models \mathsf{hyps}(\rho)$. So, by part (2) of the definition of rule-matching bisimulation, we have that $\sigma'' \models \mathsf{hyps}(J)$.

As $\mathsf{hyps}(J) = \bigvee_{\rho' \in J} \mathsf{hyps}(\rho')$, we have $\sigma'' \models \mathsf{hyps}(\rho')$ for some

$$\rho' = \frac{H'}{Q \xrightarrow{a} Q'} \in J.$$

As $\sigma'' \models \mathsf{hyps}(\rho')$, there is a substitution τ' with $\tau' = \sigma''$ on $\mathsf{vars}(Q)$ such that $\tau' \models H'$. (Note that τ' need not be consistent with σ'' on $\mathsf{TV}(\rho)$.)

We now let

$$\tau(x) = \begin{cases} \sigma''(x) & x \in \mathsf{vars}(\rho) \cup \mathsf{vars}(Q) \\ \tau'(x) & \text{otherwise.} \end{cases} \tag{13}$$

Note that $\tau = \sigma = \sigma'' = \tau'$ on $\mathsf{vars}(Q)$.

We claim that $\tau \models H'$. To see that this is indeed the case, we consider each hypothesis in H' in turn:

$x \xrightarrow{b} y$, $y \in \mathsf{TV}(\rho)$:

In this case, by Definition 5.1(1) d, we have $x \xrightarrow{b} y \in H \cap H'$: that is, the same transition formula is an antecedent of both rules. As $\sigma'' \models H$, we have $\sigma''(x) \xrightarrow{b} \sigma''(y)$. By the definition of τ, we have $\tau(x) = \sigma''(x)$ and $\tau(y) = \sigma''(y)$, so $\tau \models x \xrightarrow{b} y$.

$x \xrightarrow{b} y$, $y \notin \mathsf{TV}(\rho)$:

In this case, $x \in \mathsf{SV}(\rho') = \mathsf{vars}(Q)$, so $\tau(x) = \tau'(x)$. As $y \notin \mathsf{vars}(\rho) \cup \mathsf{vars}(Q)$, we have that $\tau(y) = \tau'(y)$. Since, by construction, $\tau' \models H'$, we have $\tau(x) \equiv \tau'(x) \xrightarrow{b} \tau'(y) \equiv \tau(y)$, and thus $\langle \tau \models x \xrightarrow{b} y \rangle$, as desired.

$x \overset{b}{\nrightarrow}$:

In this case, we have $x \in \mathsf{SV}(\rho') = \mathsf{vars}(Q)$, so the substitutions τ and τ' give the same value for x. Since $\tau' \models H'$, we have $\tau'(x) \overset{b}{\nrightarrow}$, so $\tau(x) \overset{b}{\nrightarrow}$, as desired.

So we have $\tau \models H'$, and thus ρ' fires on τ. That is, we have $Q\sigma \equiv Q\tau \xrightarrow{a} Q'\tau$. We claim that $Q'\tau$ is the closed term q we were looking for. In fact, $P'\sigma' \equiv P'\tau$, as $\sigma' = \sigma'' = \tau$ on $\mathsf{vars}(\rho)$, which shows that $\sim$ is a bisimulation relation. □

Appendix B. Proof of Theorem 7.2

Proof. The 'if' direction follows from Theorem 5.3.

https://doi.org/10.1017/S0960129511000417 Published online by Cambridge University Press

To prove that $\mathsf{Bisim}(G) \models P = Q$ implies $P \underline{\leftrightarrow}^{RM} Q$, it is sufficient to show that the relation

$$\approx = \{(P, Q) \mid \mathsf{Bisim}(G) \models P = Q, \quad P, Q \text{ persistent}\}$$

is a rule-matching bisimulation. To this end, let P, Q be persistent contexts such that $\mathsf{Bisim}(G) \models P = Q$, and let

$$\rho = \frac{H}{P \xrightarrow{a} P} \in R_G(P).$$

By Theorem 3.2, we may safely assume that ρ is such that $\mathsf{TV}(\rho) \cap \mathsf{vars}(Q) = \emptyset$. We want to find a finite set J_ρ of valid ruloids for Q satisfying the conditions of the definition of $\underline{\leftrightarrow}^{RM}$. We will now show how to construct such a J_ρ.

Let σ be a closed substitution such that $\rightarrow_G, \sigma \models H$. Then, since ρ is sound, we have $P\sigma \xrightarrow{a} P\sigma$, and since $\mathsf{Bisim}(G) \models P = Q$, we have $P\sigma \underline{\leftrightarrow} Q\sigma$. So there exists a process q such that $Q\sigma \xrightarrow{a} q$ and $P\sigma \underline{\leftrightarrow} q$. By Theorem 3.2, this is because there exists a ruloid

$$\rho'_\sigma = \frac{H'}{Q \xrightarrow{a} Q} \in R_G(Q)$$

and a substitution τ_σ such that:

— $\tau_\sigma \models H'$; and
— $Q\sigma \equiv Q\tau_\sigma \equiv q$.

Note that $Q\sigma \equiv Q\tau_\sigma$ implies that $\sigma = \tau_\sigma$ on $\mathsf{vars}(Q)$. By suitably renaming the variables in $\mathsf{TV}(\rho'_\sigma)$, it is now easy to construct a valid ruloid

$$\hat{\rho}_\sigma = \frac{\hat{H}'}{Q \xrightarrow{a} Q}$$

for Q, and a modified substitution $\hat{\tau}_\sigma$ such that:

— $\mathsf{TV}(\hat{\rho}_\sigma) \cap \mathsf{TV}(\rho) = \emptyset$ and $\mathsf{TV}(\hat{\rho}_\sigma) \cap \mathsf{vars}(P) = \emptyset$;
— $\hat{\tau}_\sigma \models \hat{H}'$; and
— $Q\sigma \equiv Q\tau_\sigma = Q\hat{\tau}_\sigma$.

Thus, for each closed substitution σ such that $\rightarrow_G, \sigma \models H$, we can construct a valid ruloid $\hat{\rho}_\sigma$ and a closed substitution $\hat{\tau}_\sigma$ with the above properties.

We now take $J_\rho = \{\hat{\rho}_\sigma \mid \rightarrow_G, \sigma \models H\}$. Note that by the Ruloid Theorem, we can assume without loss of generality that J_ρ is finite. We claim that J_ρ is a set of ruloids matching the conditions in Definition 5.1. In fact, each $\hat{\rho}_\sigma$ has action a, $P \approx Q$ by construction, and the conditions Definition 5.1 (1) c and (1) d are trivially met by construction. Moreover, we have $\models \mathsf{hyps}(\rho) \Rightarrow \mathsf{hyps}\left(J_\rho\right)$. We assume, in fact, that $\rightarrow_G, \sigma \models \mathsf{hyps}(\rho)$. Then we know that, by construction, $\hat{\tau}_\sigma \models \mathsf{hyps}(\hat{\rho}_\sigma)$, and that $\sigma = \hat{\tau}_\sigma$ on $\mathsf{vars}(Q)$. Since $\mathsf{vars}\left(\mathsf{hyps}\left(\hat{\rho}_\sigma\right)\right) \subseteq \mathsf{vars}(Q)$, Lemma A.2 gives $\sigma \models \mathsf{hyps}(\hat{\rho}_\sigma)$. And then since $\hat{\rho}_\sigma \in J_\rho$, we have $\sigma \models \mathsf{hyps}\left(J_\rho\right)$, and thus $\approx$ is indeed a rule-matching bisimulation. □

Appendix C. Proof of Theorem 7.4

Proof. Let G be a non-inheriting GSOS language that for each $P \in \mathbb{T}(\Sigma_G)$ and $c \in \mathsf{Act}$ contains at most one ruloid for P having $c \in \mathsf{Act}$ as action. Let G' be the disjoint extension

https://doi.org/10.1017/S0960129511000417 Published online by Cambridge University Press

of G obtained by adding to G the operations and rules of the language BCCSP (van Glabbeek 2001; Milner 1989) with Act as its set of actions. Let P and Q be terms over Σ_G, and assume that $\mathsf{Bisim}(G') \models P = Q$. We aim to show that $P \underline{\leftrightarrow}_{G'}^{RM} Q$. To prove our claim, it suffices to show that the relation

$$\approx = \{(P, Q) \mid \mathsf{Bisim}(G') \models P = Q,\ P, Q \in \mathbb{T}(\Sigma_G)\}$$

is a rule-matching bisimulation. We first note that $\approx$ is symmetric since $\underline{\leftrightarrow}_{G'}$ is.

We now assume that $P \underline{\leftrightarrow}_{G'} Q$ and $P, Q \in \mathbb{T}(\Sigma_G)$. We proceed to argue that the conditions in Definition 5.1 are met by $\approx$. Using Theorem 3.2, we start by constructing the set of ruloids for P and Q in such a way that $\big(\mathsf{TV}(\rho') \cup \mathsf{TV}(\rho)\big) \cap \big(\mathsf{SV}(\rho) \cup \mathsf{SV}(\rho')\big) = \emptyset$. This satisfies the condition Definition 5.1 (1) c.

Let

$$\rho = \frac{H}{P \xrightarrow{a} P'}$$

be a ruloid for P, which we may assume is not junk. We will now show that there is a ruloid

$$\rho' = \frac{H'}{Q \xrightarrow{a} Q'}$$

such that:

(1) $P' \approx Q'$.

(2) If $y \in \mathsf{TV}(\rho) \cap \mathsf{TV}(\rho')$, then $x \xrightarrow{b} y \in H \cap H'$ for some source variable $x \in \mathsf{SV}(\rho) \cap \mathsf{SV}(\rho')$ and action b.

(3) $\models_{G'} \mathsf{hyps}(\rho) \Rightarrow \mathsf{hyps}(\rho')$.

We first note that by the proviso of the theorem, the set of ruloids for Q with action a must be a singleton. Indeed, that set cannot be empty since ρ is a ruloid for P whose premises can be satisfied by some closed substitution σ, since ρ is not junk. This means that $P\sigma$ affords an a-labelled transition, but $Q\sigma$ would not. This contradicts our assumption that $P \underline{\leftrightarrow}_{G'} Q$.

Let

$$\rho' = \frac{H'}{Q \xrightarrow{a} Q'}$$

be the only ruloid for Q with action a. We will now establish each of the conditions (1)–(3) in turn:

(1) To show that $P' \approx Q'$, we begin by observing that $P', Q' \in \mathbb{T}(\Sigma_G)$ because G' is a disjoint extension of G and $P, Q \in \mathbb{T}(\Sigma_G)$. We are therefore left to prove that $P' \underline{\leftrightarrow}_{G'} Q'$. In order to show a contradiction, we assume that $P' \underline{\nleftrightarrow}_{G'} Q'$. This means that there is a closed substitution σ', mapping variables to terms in $\mathrm{T}(\Sigma_{G'})$, such that $P'\sigma' \underline{\nleftrightarrow}_{G'} Q'\sigma'$. We will now use σ' to construct a closed substitution σ such that $P\sigma \underline{\nleftrightarrow}_{G'} Q\sigma$, the existence of which will contradict our assumption that $\mathsf{Bisim}(G') \models P = Q$.

https://doi.org/10.1017/S0960129511000417 Published online by Cambridge University Press

We define σ by

$$\sigma(x) = \begin{cases} \sum \left\{ b.\sigma'(x') \mid (x \xrightarrow{b} x') \in H \right\} & \text{if } x \in \mathsf{vars}(P) \\ \sigma'(x) & \text{otherwise.} \end{cases}$$

Note that terms of the form $\sum\{b.\sigma'(x') \mid (x \xrightarrow{b} x') \in H\}$ are in $\mathrm{T}(\Sigma_{G'})$ since $\Sigma_{G'}$ includes the signature of BCCSP. As usual, an empty sum stands for $\mathbf{0}$.

We claim that $\rightarrow_{G'}, \sigma \models H$. Indeed, suppose $(x \xrightarrow{b} x') \in H$. Hence, since $x \notin \mathsf{vars}(P)$, we have

$$\sigma(x) \xrightarrow{b} \sigma'(x') = \sigma(x').$$

We now assume $x \overset{b}{\nrightarrow} \in H$. Since ρ is not junk, H contains no formula of the form $(x \xrightarrow{b} x')$. Therefore, by definition, $\sigma(x) \overset{b}{\nrightarrow}$. It follows that ruloid ρ fires under substitution σ, so

$$P\sigma \xrightarrow{a} P'\sigma = P'\sigma'.$$

(The equality $P'\sigma = P'\sigma'$ holds because G is non-inheriting and therefore no variable occurring in P' also occurs in P.) Since $P\sigma \leftrightarrows_{G'} Q\sigma$ and ρ' is the only a-labelled ruloid for Q, there is a transition

$$Q\sigma \xrightarrow{a} Q'\sigma = Q'\sigma' \leftrightarrows_{G'} P'\sigma'.$$

(Again, the equality $Q'\sigma = Q'\sigma'$ holds because G is non-inheriting and no target variable occurring in ρ' also occurs in Q.) This contradicts our assumption that $P'\sigma' \overset{\leftrightarrow}{\nrightarrow}_{G'} Q'\sigma'$. We may therefore conclude that $P' \leftrightarrows_{G'} Q'$, as desired.

(2) We need to show that if $y \in \mathsf{TV}(\rho) \cap \mathsf{TV}(\rho')$, then $x \xrightarrow{b} y \in H \cap H'$ for some source variable $x \in \mathsf{SV}(\rho) \cap \mathsf{SV}(\rho')$ and action b.

Assume that $y \in \mathsf{TV}(\rho) \cap \mathsf{TV}(\rho')$. Let $x \xrightarrow{b} y$ and $z \xrightarrow{c} y$ be the only premises with target y in H and H', respectively. We will now show that $b = c$ and $x = z$. In order to show a contradiction, we assume first that $x \neq z$. Using this assumption, we shall construct a closed substitution σ such that $P\sigma \overset{\leftrightarrow}{\nrightarrow}_{G'} Q\sigma$, which contradicts $\mathsf{Bisim}(G') \models P = Q$. Let X be the collection of target variables w' in ρ and ρ' that are different from y and for which there is a premise of the form $w \xrightarrow{c} w'$ in $H \cup H'$. For each $w \in \mathsf{vars}(P)$ and action d, let the closed term p_w^d be given by

$$p_w^d = \begin{cases} d.d.\mathbf{0} & \exists w'.\ (w \xrightarrow{d} w') \in H \\ \mathbf{0} & \text{otherwise.} \end{cases} \tag{14}$$

We define σ by

$$\sigma(w) = \begin{cases} \sum \left\{ d.\mathbf{0} \mid \exists w'.\ (w \xrightarrow{d} w') \in H \right\} + p_w^c & \text{if } w \in \mathsf{vars}(P) - \{z\} \\ \sum \left\{ d.\mathbf{0} \mid d \neq c \wedge \exists w'.\ (z \xrightarrow{d} w') \in H \right\} + c.c.\mathbf{0} & \text{if } w = z \\ c.\mathbf{0} & \text{if } w \in X, \text{ and} \\ \mathbf{0} & \text{otherwise.} \end{cases}$$

https://doi.org/10.1017/S0960129511000417 Published online by Cambridge University Press

It is not hard to see that σ satisfies H, but not the formula $z \xrightarrow{c} y$. Therefore, ruloid ρ fires under substitution σ, but ρ' does not. Reasoning as above, we can show, as desired, that $P\sigma$ and $Q\sigma$ are not bisimilar.

So, $x \xrightarrow{b} y \in H$ and $x \xrightarrow{c} y \in H'$. We shall now argue that $b = c$, completing the proof for this case. In order to show a contradiction, we assume that $b \neq c$. As above, using this assumption, we shall construct a closed substitution σ such that $P\sigma \mathrel{\underline{\leftrightarrow}_{G'}} Q\sigma$, which contradicts $\mathsf{Bisim}(G') \models P = Q$.

Let Y be the collection of target variables w' in ρ for which there is a premise of the form $w \xrightarrow{b} w'$ in H. In particular, $y \in Y$. We define σ as follows, where the term p^b_w is defined as in (14):

$$\sigma(w) = \begin{cases} \sum \left\{ d.\mathbf{0} \mid d \neq b \wedge \exists w'.\ (w \xrightarrow{d} w') \in H \right\} + p^b_w & \text{if } w \in \mathsf{vars}(P) \\ b.\mathbf{0} & \text{if } w \in Y \\ \mathbf{0} & \text{otherwise.} \end{cases}$$

It is not hard to see that σ satisfies H, but not the formula $x \xrightarrow{c} y$. Therefore, ruloid ρ fires under substitution σ, but ρ' does not. Reasoning as above, we can show, as desired, that $P\sigma$ and $Q\sigma$ are not bisimilar. We may now finally conclude that $b = c$ and therefore that $x \xrightarrow{b} y \in H \cap H'$, which is what we set out to show.

(3) To show that $\models_{G'} \mathsf{hyps}(\rho) \Rightarrow \mathsf{hyps}(\rho')$, we assume, in order to show a contradiction, that there is a closed substitution σ that satisfies $\mathsf{hyps}(\rho)$ but not $\mathsf{hyps}(\rho')$. We shall use σ to construct a substitution σ' such that $P\sigma' \mathrel{\underline{\leftrightarrow}_{G'}} Q\sigma'$, which contradicts our assumption that $P \mathrel{\underline{\leftrightarrow}_{G'}} Q$.

We define σ' by

$$\sigma'(x) = \begin{cases} \sum \left\{ b.\mathbf{0} \mid \exists p.\ \sigma(x) \xrightarrow{b} p \right\} & \text{if } x \in \mathsf{SV}(\rho) \cup \mathsf{SV}(\rho') \\ \mathbf{0} & \text{otherwise.} \end{cases}$$

We claim that $\rightarrow_{G'}, \sigma \models H$. So we suppose $(x \xrightarrow{b} x') \in H$. Then, $x \xrightarrow{b}$ is a conjunct of $\mathsf{hyps}(\rho)$ and $x \in \mathsf{SV}(\rho)$. Since σ satisfies $\mathsf{hyps}(\rho)$, it follows that $\sigma(x) \xrightarrow{b} p$ for some closed term p. By the definition of σ' and the fact that $x' \notin \mathsf{SV}(\rho) \cup \mathsf{SV}(\rho')$, we have

$$\sigma'(x) \xrightarrow{b} \mathbf{0} = \sigma'(x').$$

We now assume $x \xrightarrow{b}\in H$. Then, $x \xrightarrow{b}$ is a conjunct of $\mathsf{hyps}(\rho)$ and $x \in \mathsf{SV}(\rho)$. Since σ satisfies $\mathsf{hyps}(\rho)$, it follows that $\sigma(x) \xrightarrow{b}$. By the definition of σ', we have $\sigma'(x) \xrightarrow{b}$, and we are done. It follows that ruloid ρ fires under substitution σ', so

$$P\sigma' \xrightarrow{a} P'\sigma'.$$

We shall now argue that $\rightarrow_{G'}, \sigma' \not\models H'$. This means that $Q\sigma'$ does not afford an a-labelled transition, so $P\sigma' \mathrel{\underline{\leftrightarrow}_{G'}} Q\sigma'$, as desired.

https://doi.org/10.1017/S0960129511000417 Published online by Cambridge University Press

Since σ does not satisfy $\mathsf{hyps}(\rho')$, there is a conjunct of that formula that is not satisfied by σ. We proceed with the proof of our claim by considering the two possible forms this conjunct may take:

— Assume that σ does not satisfy a conjunct of the form $x \xrightarrow{b}$ in $\mathsf{hyps}(\rho')$.
This means that $\sigma(x) \overset{b}{\nrightarrow}$ and that $(x \xrightarrow{b} x') \in H'$ for some x'. Note that $x \in \mathsf{SV}(\rho')$. Therefore $\sigma'(x) \overset{b}{\nrightarrow}$, by the definition of σ', and $\rightarrow_{G'}, \sigma' \not\models H'$, as claimed.

— Assume that σ does not satisfy a conjunct of the form $x \overset{b}{\nrightarrow}$ in $\mathsf{hyps}(\rho')$.
This means that $\sigma(x) \xrightarrow{b} p$ for some closed term p and that $(x \overset{b}{\nrightarrow}) \in H'$. Note that $x \in \mathsf{SV}(\rho')$. Therefore $\sigma'(x) \xrightarrow{b} \mathbf{0}$, by the definition of σ', and $\rightarrow_{G'}, \sigma' \not\models H'$, as claimed.

This completes the proof. □

Acknowledgements

We would like to thank the anonymous referees for their constructive remarks, which led to improvements in the paper.

References

Aceto, L., Birgisson, A., Ingolfsdottir, A., Mousavi, M. and Reniers, M. (2010) Rule formats for determinism and idempotence. In: Arbab, F. and Sirjani, M. (eds.) Fundamentals of Software Engineering, Third IPM International Conference, FSEN 2009, Revised Selected Papers. *Springer-Verlag Lecture Notes in Computer Science* **5961** 146–161.

Aceto, L., Bloom, B. and Vaandrager, F. (1994) Turning SOS rules into equations. *Information and Computation* **111** (1) 1–52.

Aceto, L., Cimini, M. and Ingolfsdottir, A. (2010a) A bisimulation-based method for proving the validity of equations in GSOS languages. In: Klin, B. and Sobocinski, P. (eds.) Proceedings Sixth Workshop on Structural Operational Semantics (SOS 2009). *Electronic Proceedings in Theoretical Computer Science* **18** 1–16.

Aceto, L., Fokkink, W., Ingolfsdottir, A. and Luttik, B. (2005) Finite equational bases in process algebra: Results and open questions. In: Middeldorp, A., van Oostrom, V., van Raamsdonk, F. and de Vrijer, R. C. (eds.) Processes, Terms and Cycles: Steps on the Road to Infinity, Essays Dedicated to Jan Willem Klop, on the Occasion of His 60th Birthday. *Springer-Verlag Lecture Notes in Computer Science* **3838** 338–367.

Aceto, L., Fokkink, W., Ingolfsdottir, A. and Luttik, B. (2009) A finite equational base for CCS with left merge and communication merge. *ACM Transactions on Compututational Logic* **10** (1).

Aceto, L., Fokkink, W. and Verhoef, C. (2001) Structural operational semantics. In: Bergstra, J., Ponse, A. and Smolka, S. A. (eds.) *Handbook of Process Algebra*, Elsevier 197–292.

Aceto, L., Ingolfsdottir, A., Luttik, B. and van Tilburg, P. (2008) Finite equational bases for fragments of CCS with restriction and relabelling. In: Ausiello, G., Karhumäki, J., Mauri, G. and Ong, C.-H. L. (eds.) Fifth IFIP International Conference On Theoretical Computer Science – TCS 2008. *IFIP* **273** 317–332.

Aceto, L., Ingolfsdottir, A., Mousavi, M. and Reniers, M. (2010b) A rule format for unit elements. In: van Leeuwen, J., Muscholl, A., Peleg, D., Pokorný, J. and Rumpe, B. (eds.) Proceedings

https://doi.org/10.1017/S0960129511000417 Published online by Cambridge University Press

SOFSEM 2010: Theory and Practice of Computer Science, 36th Conference on Current Trends in Theory and Practice of Computer Science. *Springer-Verlag Lecture Notes in Computer Science* **5901** 141–152.

Baeten, J. and Bergstra, J. (1990) Process algebra with a zero object. In: Baeten, J. C. M. and Klop, J. W. (eds.) Proceedings CONCUR 90. *Springer-Verlag Lecture Notes in Computer Science* **458** 83–98.

Baeten, J. and de Vink, E. P. (2004) Axiomatizing GSOS with termination. *Journal of Logic and Algebraic Programming* **60-61** 323–351.

Baeten, J. and Vaandrager, F. (1992) An algebra for process creation. *Acta Informatica* **29** (4) 303–334.

Baeten, J. and Weijland, P. (1990) *Process Algebra*, Cambridge Tracts in Theoretical Computer Science **18**, Cambridge University Press.

Bloom, B., Fokkink, W. and van Glabbeek, R. (2004) Precongruence formats for decorated trace semantics. *ACM Transactions on Computational Logic* **5** (1) 26–78.

Bloom, B., Istrail, S. and Meyer, A. (1995) Bisimulation can't be traced. *Journal of the ACM* **42** (1) 232–268.

Bruni, R., de Frutos-Escrig, D., Martí-Oliet, N. and Montanari, U. (2000) Bisimilarity congruences for open terms and term graphs via tile logic. In: Palamidessi, C. (ed.) CONCUR 2000 – Concurrency Theory, 11th International Conference, Proceedings. *Springer-Verlag Lecture Notes in Computer Science* **1877** 259–274.

Cranen, S., Mousavi, M. and Reniers, M. (2008) A rule format for associativity. In: van Breugel, F. and Chechik, M. (eds.) Proceedings of CONCUR 2008 – Concurrency Theory, 19th International Conference. *Springer-Verlag Lecture Notes in Computer Science* **5201** 447–461.

Doumenc, G., Madelaine, E. and de Simone, R. (1990) Proving process calculi translations in ECRINS. Technical Report RR1192, INRIA.

Fokkink, W., van Glabbeek, R. and de Wind, P. (2006) Compositionality of Hennessy–Milner logic by structural operational semantics. *Theoretical Computer Science* **354** (3) 421–440.

Fokkink, W. and Verhoef, C. (1998) A conservative look at operational semantics with variable binding. *Information and Computation* **146** (1) 24–54.

van Glabbeek, R. (2001) The linear time–branching time spectrum. I. The semantics of concrete, sequential processes. In: Bergstra, J., Ponse, A. and Smolka, S. A. (eds.) *Handbook of Process Algebra*, Elsevier 3–99.

Groote, J. F. and Vaandrager, F. (1992) Structured operational semantics and bisimulation as a congruence. *Information and Computation* **100** (2) 202–260.

Hennessy, M. (1988) *Algebraic Theory of Processes*, MIT Press.

Hennessy, M. and Milner, R. (1985) Algebraic laws for nondeterminism and concurrency. *Journal of the ACM* **32** (1) 137–161.

Hoare, C. A. R. (1985) *Communicating Sequential Processes*. Prentice-Hall International. (Available at `http://www.usingcsp.com/cspbook.pdf`.)

Hoare, C. A. R. *et al.* (1987) Laws of programming. *Communications of the ACM* **30** (8) 672–686.

Larsen, K. G. and Liu, X. (1991) Compositionality through an operational semantics of contexts. *Journal of Logic and Computation* **1** (6) 761–795.

Madelaine, E. and Vergamini, D. (1991) Finiteness conditions and structural construction of automata for all process algebras. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science* **3** 275–292.

Milner, R. (1984) A complete inference system for a class of regular behaviours. *Journal of Computer and System Sciences* **28** 439–466.

https://doi.org/10.1017/S0960129511000417 Published online by Cambridge University Press

Milner, R. (1989) *Communication and Concurrency*, Prentice-Hall International.

Mousavi, M. and Reniers, M. (2005) Orthogonal extensions in structural operational semantics. In: Proceedings of the 32nd International Colloquium on Automata, Languages and Programming (ICALP'05). *Springer-Verlag Lecture Notes in Computer Science* **3580** 1214–1225.

Mousavi, M., Reniers, M. and Groote, J. F. (2005) A syntactic commutativity format for SOS. *Information Processing Letters* **93** 217–223.

Mousavi, M., Reniers, M. and Groote, J. F. (2007) SOS formats and meta-theory: 20 years after. *Theoretical Computer Science* **373** (3) 238–272.

Park, D. (1981) Concurrency and automata on infinite sequences. In: Deussen, P. (ed.) 5th GI Conference, Karlsruhe, Germany. *Springer-Verlag Lecture Notes in Computer Science* **104** 167–183.

Plotkin, G. D. (1981) A structural approach to operational semantics. Report DAIMI FN-19, Computer Science Department, Aarhus University.

Plotkin, G. D. (2004) A structural approach to operational semantics. *Journal of Logic and Algebraic Programming* **60-61** 17–139. (This is a revised version of Plotkin (1981).)

Rensink, A. (2000) Bisimilarity of open terms. *Information and Computation* **156** (1-2) 345–385.

de Simone, R. (1984) *Calculabilité et Expressivité dans l'Algèbre de Processus Parallèles* MEIJE, Thèse de 3^e cycle, Univ. Paris 7.

de Simone, R. (1985) Higher-level synchronising devices in MEIJE–SCCS. *Theoretical Computer Science* **37** 245–267.

van Weerdenburg, M. (2008) Automating soundness proofs. In M. Hennessy and B. Klin (eds.) Proceedings of the Workshop on Structural Operational Semantics (SOS 2008). *Electronic Notes in Theoretical Computer Science* **229** (4) 107–118.

https://doi.org/10.1017/S0960129511000417 Published online by Cambridge University Press