Axiomatizing GSOS with Predicates*

Luca Aceto[1], Georgiana Caltais[1], Eugen-Ioan Goriac[1], and Anna Ingolfsdottir[1]
[luca,gcaltais10,egoriac10,annai]@ru.is

ICE-TCS, School of Computer Science, Reykjavik University, Iceland

Abstract. In this paper, we introduce an extension of the GSOS rule format with predicates such as termination, convergence and divergence. For a restriction of this format to what we call explicit predicates, we generalize the technique proposed by Aceto, Bloom and Vaandrager for the automatic generation of ground-complete axiomatizations of bisimilarity over GSOS systems. Our procedure is implemented in a tool that receives SOS specifications as input and derives the corresponding axiomatizations automatically. This paves the way to checking strong bisimilarity over process terms by means of theorem-proving techniques. We also describe how the proposed techniques and their implementation can be used to handle more general predicates than the explicit ones.

1 Introduction

One of the greatest challenges in computer science is the development of rigorous methods for the specification and verification of reactive systems, *i.e.*, systems that compute by interacting with their environment. Typical examples include embedded systems, control programs and distributed communication protocols. Over the last three decades, process algebras, such as ACP [3], CCS [12] and CSP [10], have been successfully used as common languages for the description of both actual systems and their specifications. In this context, verifying whether the implementation of a reactive system complies to its specification reduces to proving that the corresponding process terms are related by some notion of behavioural equivalence or preorder [15].

One approach to proving equivalence between two terms is to exploit the equational style of reasoning supported by process algebras. In this approach, one obtains a (ground-)complete axiomatization of the behavioural relation of interest and uses it to prove the equivalence between the terms describing the specification and the implementation by means of equational reasoning, possibly in conjunction with proof rules to handle recursively-defined process specifications.

* The authors have been been partially supported by the projects "New Developments in Operational Semantics" (nr. 080039021) and "Meta-theory of Algebraic Process Theories" (nr. 100014021) of the Icelandic Fund for Research. The work on the paper was partly carried out while Luca Aceto and Anna Ingolfsdottir held an Abel Extraordinary Chair at Universidad Complutense de Madrid, Spain, supported by the NILS Mobility Project.

Finding a “finitely specified”, (ground-)complete axiomatization of a behavioural equivalence over a process algebra is often a highly non-trivial task. However, as shown in [2] in the setting of bisimilarity [12, 13], this process can be automated for process languages with an operational semantics given in terms of rules in the GSOS format of Bloom, Istrail and Meyer [7]. In that reference, Aceto, Bloom and Vaandrager provided an algorithm that, given a GSOS language as input, produces as output a “conservative extension” of the original language with auxiliary operators together with a finite axiom system that is sound and ground-complete with respect to bisimilarity (see, *e.g.*, [14, 11, 1, 9] for further results in this line of research). As the operational specification of several operators often requires a clear distinction between successful termination and deadlock, an extension of the above-mentioned approach to the setting of GSOS with a predicate for termination was proposed in [4].

In this paper we contribute to the line of the work in [2] and [4]. Inspired by [4], we introduce the *preg* rule format, a natural extension of the GSOS format with an arbitrary collection of predicates such as termination, convergence and divergence. We further adapt the theory in [2] to this setting and give a procedure for obtaining ground-complete axiomatizations for bisimilarity over *epreg* systems. (The notation *epreg* stands for the restriction of the *preg* rule format to a class of predicates that we call *explicit predicates.*) More specifically, we develop a general procedure that, given an *epreg* language as input, automatically synthesizes a conservative extension of that language and a finite axiom system that, in conjunction with an infinitary proof rule, yields a sound and ground-complete axiomatization of bisimilarity over the extended language. The work we present in this paper is based on the one reported in [2, 4]. However, handling more general predicates than immediate termination requires the introduction of some novel technical ideas. In particular, the problem of axiomatizing bisimilarity over an *epreg* language is reduced to that of axiomatizing the relation over finite trees whose nodes may be labelled with predicates. In order to do so, one needs to take special care in axiomatizing negative premises in rules that may have positive and negative premises involving predicates and transitions.

The results of the current paper have been used for the implementation of a Maude [8] tool[1] that enables the user to specify *epreg* systems in a uniform fashion, and that automatically derives the associated axiomatizations. This paves the way to checking bisimilarity over process terms by means of theorem-proving techniques for a large class of systems that can be expressed using *epreg* language specifications.

Paper structure. In Section 2 we introduce the *preg* rule format. In Section 3 we introduce an appropriate “core” language for expressing finite trees with predicates. We also provide a ground-complete axiomatization for bisimilarity over this type of trees, as our aim is to prove the completeness of our final axiomatization by head normalizing general *preg* terms, and therefore by reducing the completeness problem for arbitrary languages to that for trees.

[1] available at `http://goriac.info/tools/preg-axiomatizer/`

Head normalizing general *preg* terms is not a straightforward process. Therefore, following [2], in Section 4 we introduce the notion of smooth and distinctive operation, adapted to the current setting. These operations are designed to "capture the behaviour of general *preg* operations", and are defined by rules satisfying a series of syntactic constraints with the purpose of enabling the construction of head normalizing axiomatizations. Such axiomatizations are based on a collection of equations that describe the interplay between smooth and distinctive operations, and the operations in the signature for finite trees. For our case, the limitation of these operations is that they can only be used for head normalizing terms built using operations defined by rules matching the *epreg* format. The existence of a sound and ground-complete axiomatization characterizing the bisimilarity of *epreg* processes is finally proven in Section 5. In Section 6 we draw some conclusions and provide pointers to future work. Proofs of technical results, additional material as well as some discussion of the design choices we made in the development of the work presented in this paper may be found in the appendices.

2 GSOS with predicates

In this section we present the *preg* systems which are a generalization of GSOS [7] systems.

Consider a countably infinite set V of *process variables* (usually denoted by x, y, z) and a signature Σ consisting of a set of *operations* (denoted by f, g). The set of *process terms* $\mathbb{T}(\Sigma)$ is inductively defined as follows: each variable $x \in \mathit{Var}$ is a term; if $f \in \Sigma$ is an operation of arity l, and if $S_1, \ldots, S_l$ are terms, then $f(S_1, \ldots, S_l)$ is a term. We write $T(\Sigma)$ in order to represent the set of *closed process terms* (*i.e.*, terms that do not contain variables), ranged over by t, s. A *substitution* σ is a function of type $V \to \mathbb{T}(\Sigma)$. If the range of a substitution is included in $T(\Sigma)$, we say that it is a *closed substitution.* Moreover, we write $[x \mapsto t]$ to represent a substitution that maps the variable x to the term t. Let $\vec{x} = x_1, \ldots, x_n$ be a sequence of pairwise distinct variables. A Σ*-context* $C[\vec{x}]$ is a term in which at most the variables $\vec{x}$ appear.

Let $\mathcal{A}$ be a finite, nonempty set of *actions* (denoted by a, b, c). A *positive transition formula* is a triple (S, a, S') written $S \xrightarrow{a} S'$, with the intended meaning: process S performs action a and becomes process S'. A *negative transition formula* (S, a) written $S \overset{a}{\nrightarrow}$, states that process S cannot perform action a. Note that S, S' may contain variables. The "intended meaning" applies to closed process terms.

We now define *preg* – *pr*edicates *e*xtension of the *G*SOS rule format. Let $\mathcal{P}$ be a finite set of *predicates* (denoted by P, Q). A *positive predicate formula* is a pair (P, S) written PS, saying that process S satisfies predicate P. Dually, a *negative predicate formula* $\neg P\,S$ states that process S does not satisfy predicate P.

Definition 1 (*preg* rule format). *Consider $\mathcal{A}$, a set of actions, and $\mathcal{P}$, a set of predicates.*

1. *A preg* transition rule *for an l-ary operation f is a deduction rule of the form:*

$$\frac{\begin{array}{cc}\{x_i \xrightarrow{a_{ij}} y_{ij} \mid i \in I^+, j \in I_i^+\} & \{P_{ij} x_i \mid i \in J^+, j \in J_i^+\} \\ \{x_i \stackrel{b}{\nrightarrow} \mid i \in I^-, b \in \mathcal{B}_i\} & \{\neg Q x_i \mid i \in J^-, Q \in \mathcal{Q}_i\}\end{array}}{f(x_1, \ldots, x_l) \xrightarrow{c} C[\vec{x}, \vec{y}]}$$

   where

   (a) $x_1, \ldots, x_l$ are pairwise distinct variables;

   (b) $I^+, J^+, I^-, J^- \subseteq L = \{1, \ldots, l\}$ and each I_i^+ and J_i^+ is finite;

   (c) a_{ij}, b and c are actions in $\mathcal{A}$ ($\mathcal{B}_i \subseteq \mathcal{A}$); and

   (d) P_{ij} and Q are predicates in $\mathcal{P}$ ($\mathcal{Q}_i \subseteq \mathcal{P}$).

2. *A preg* predicate rule *for an l-ary operation f is a deduction rule similar to the one above, with the only difference that its conclusion has the form $P(f(x_1, \ldots, x_l))$ for some $P \in \mathcal{P}$.*

Let ρ be a *preg* (transition or predicate) rule for f. The symbol f is the *principal operation* of ρ. All the formulas above the line are *antecedents* and the formula below is the *consequent*. We say that a position i for ρ is *tested positively* if $i \in I^+ \cup J^+$ and $I^+ \cup J^+ \neq \emptyset$. Similarly, i is *tested negatively* if $i \in I^- \cup J^-$ and $\mathcal{B}_i \cup \mathcal{Q}_i \neq \emptyset$. Whenever ρ is a transition rule for f, we say that $f(\vec{x})$ is the *source*, $C[\vec{x}, \vec{y}]$ is the *target*, and c is the *action* of ρ. Whenever ρ is a predicate rule for f, we call $f(\vec{x})$ the *test* of ρ.

In order to avoid confusion, if in a certain context we use more than one rule, e.g. ρ, ρ', we parameterize the corresponding sets of indices with the name of the rule, *e.g.*, I_ρ^+, $J_{\rho'}^-$.

Definition 2 (*preg* system). *A* preg *system is a pair $G = (\Sigma_G, \mathcal{R}_G)$, where Σ_G is a finite signature and $\mathcal{R}_G = \mathcal{R}_G^{\mathcal{A}} \cup \mathcal{R}_G^{\mathcal{P}}$ is a finite set of* preg *rules over Σ_G ($\mathcal{R}_G^{\mathcal{A}}$ and $\mathcal{R}_G^{\mathcal{P}}$ represent the transition and, respectively, the predicate rules of G).*

Consider a *preg* system G. Formally, the operational semantics of a closed process term in G is fully characterized by the relations $\rightarrow_G \subseteq T(\Sigma_G) \times \mathcal{A} \times T(\Sigma_G)$ and $\ltimes_G \subseteq \mathcal{P} \times T(\Sigma_G)$, called the (unique) *sound and supported* transition and, respectively, predicate relations. Intuitively, soundness guarantees that $\rightarrow_G$ and $\ltimes_G$ are closed with respect to the application of the rules in $\mathcal{R}_G$ on $T(\Sigma_G)$, *i.e.*, $\rightarrow_G$ (resp. $\ltimes_G$) includes the set of all possible transitions (resp. predicates) process terms in $T(\Sigma_G)$ can perform (resp. satisfy) according to $\mathcal{R}_G$. The requirement that $\rightarrow_G$ and $\ltimes_G$ be supported means that all the transitions performed (resp. all the predicates satisfied) by a certain process term can be "derived" from the deductive system described by $\mathcal{R}_G$. As a notational convention, we write $S \xrightarrow{a}_G S'$ and $P_G S$ whenever $(S, a, S') \in \rightarrow_G$ and $(P, S) \in \ltimes_G$. We omit the subscript G when it is clear from the context. The formal definitions of $\rightarrow_G$ and $\ltimes_G$ are provided in Appendix B.

Lemma 1. *Let G be a* preg *system. Then, for each $t \in T(\Sigma_G)$ the set $\{(a, t') \mid t \xrightarrow{a} t',\ a \in \mathcal{A}\}$ is finite.*

Next we introduce the notion of *bisimilarity* – the equivalence among processes we consider in this paper.

Definition 3 (Bisimulation). *Consider a* preg *system $G = (\Sigma_G, \mathcal{R}_G)$. A symmetric relation $R \subseteq T(\Sigma_G) \times T(\Sigma_G)$ is a* bisimulation *iff:*

1. *for all $s, t, s' \in T(\Sigma_G)$, whenever $(s, t) \in R$ and $s \xrightarrow{a} s'$ for some $a \in \mathcal{A}$, then there is some $t' \in T(\Sigma_G)$ such that $t \xrightarrow{a} t'$ and $(s', t') \in R$;*
2. *whenever $(s, t) \in R$ and Ps $(P \in \mathcal{P})$ then Pt.*

Two closed terms s and t are bisimilar *$(s \sim t)$ iff there is a bisimulation relation R such that $(s, t) \in R$.*

Proposition 1. *Let G be a* preg *system. Then $\sim$ is an equivalence relation and a congruence for all operations f of G.*

Definition 4 (Disjoint extension). *A* preg *system G' is a disjoint extension of a* preg *system G, written $G \sqsubseteq G'$, if the signature and the rules of G' include those of G, and G' does not introduce new rules for operations in G.*

It is well known that if $G \sqsubseteq G'$ then two terms in $T(\Sigma_G)$ are bisimilar in G if and only if they are bisimilar in G'.

From this point forward, our focus is to find a *sound and ground-complete axiomatization of bisimilarity on closed terms* for an arbitrary *preg* system G, *i.e.*, to identify a (finite) axiom system E_G so that $E_G \vdash s = t$ *iff* $s \sim t$ for all $s, t \in T(\Sigma_G)$. The method we apply is an adaptation of the technique in [2] to the *preg* setting. The strategy is to incrementally build a finite, head-normalizing axiomatization for general *preg* terms, *i.e.*, an axiomatization that, when applied recursively, reduces the completeness problem for arbitrary terms to that for synchronization trees. This way, the proof of ground-completeness for G reduces to showing the equality of closed tree terms.

3 Preliminary steps towards the axiomatization

In this section we start by identifying an appropriate language for expressing finite trees with predicates. We continue in the style of [2], by extending the language with a kind of restriction operator used for expressing the inability of a process to perform a certain action or to satisfy a given predicate. (This operator is used in the axiomatization of negative premises.) We provide the structural operational semantics of the resulting language, together with a sound and ground-complete axiomatization of bisimulation equivalence on finite trees with predicates.

3.1 Finite trees with predicates

The language for trees we use in this paper is an extension with predicates of the language BCCSP [15]. The syntax of BCCSP consists of closed terms built from a constant δ (*deadlock*), the binary operator $\_+\_$ (*nondeterministic choice*), and the unary operators $a.\_$ (*action prefix*), where a ranges over the actions in a set $\mathcal{A}$. Let $\mathcal{P}$ be a set of predicates. For each $P \in \mathcal{P}$ we consider a process constant c_P, which "witnesses" the associated predicate in the definition of a process. Intuitively, c_P stands for a process that only satisfies predicate P and offers no transition.

A finite tree term t is built according to the following grammar:

$$t ::= \delta \mid c_P \ (\forall P \in \mathcal{P}) \mid a.t \ (\forall a \in \mathcal{A}) \mid t + t. \tag{1}$$

Intuitively, δ represents a process that does not exhibit any behaviour, $s + t$ is the nondeterministic choice between the behaviours of s and t, while $a.t$ is a process that first performs action a and behaves like t afterwards. The operational semantics that captures this intuition is given by the rules of BCCSP:

$$\frac{}{a.x \xrightarrow{a} x}\ (rl_1) \qquad \frac{x \xrightarrow{a} x'}{x + y \xrightarrow{a} x'}\ (rl_2) \qquad \frac{y \xrightarrow{a} y'}{x + y \xrightarrow{a} y'}\ (rl_3)$$

Fig. 1. The semantics of BCCSP

As our goal is to extend BCCSP, the next step is to find an appropriate semantics for predicates. As can be seen in Fig. 1, action performance is determined by the shape of the terms. Consequently, we choose to define predicates in a similar fashion.

Consider a predicate P and the term $t = c_P$. As previously mentioned, the purpose of c_P is to witness the satisfiability of P. Therefore, it is natural to consider that c_P satisfies P.

Now, take for example the *immediate termination* predicate $\downarrow$. As a term $s+s'$ exhibits the behaviour of both s and s', it is suitable to state that $(s + s') \downarrow$ if $s \downarrow$ or $s' \downarrow$. Note that for a term $t = a.t'$ the statement $t \downarrow$ is in contradiction with the meaning of immediate termination, since t can only execute action a. Predicates of this kind are called *explicit predicates* in what follows.

Consider now the *eventual termination* predicate $\lightning$. In this situation, it is proper to consider that $(s+t)\lightning$ if $s\lightning$ or $t\lightning$ and, moreover, that $a.s\lightning$ if $s\lightning$. We refer to predicates such as $\lightning$ as *implicit predicates* (that range over a set $\mathcal{P}^{\mathcal{I}}$ included in $\mathcal{P}$), since their satisfiability propagates through the structure of tree terms in an implicit fashion.

The transition rules expressing the semantics of predicates are:

$$\frac{}{Pc_P}\ (rl_4) \quad \frac{Px}{P(x+y)}\ (rl_5) \quad \frac{Py}{P(x+y)}\ (rl_6) \quad \frac{Px}{P(a.x)}, \forall P \in \mathcal{P}^{\mathcal{I}} (rl_7)$$

Fig. 2. The semantics of predicates

The operational semantics of trees with predicates is given by the set of rules (rl_1)–(rl_7) illustrated in Fig. 1 and Fig. 2. For notational consistency, we make the following conventions. Let $\mathcal{A}$ be an action set and $\mathcal{P}$ a set of predicates. Σ_{FTP} represents the signature of finite trees with predicates. $T(\Sigma_{FTP})$ is the set of (closed) tree terms built over Σ_{FTP}, and $\mathcal{R}_{FTP}$ is the set of rules (rl_1)–(rl_7). Moreover, by FTP we denote the system $(\Sigma_{FTP}, \mathcal{R}_{FTP})$.

A brief description of a) the capabilities of the framework used for predicate specification; and b) the reasons for the design decisions we have made developing the framework defined above, can be found in Appendix A.

3.2 Axiomatizing finite trees

In what follows we provide a finite sound and ground-complete axiomatization (E_{FTP}) for bisimilarity over finite trees with predicates.

The axiom system E_{FTP} consists of the following axioms:

$$\begin{array}{cc} x + y = y + x \quad (A_1) & x + x = x \ (A_3) \\ (x + y) + z = x + (y + z) \ (A_2) & x + \delta = x \ (A_4) \\ \multicolumn{2}{c}{a.(x + c_P) = a.(x + c_P) + c_P, \forall P \in \mathcal{P}^{\mathcal{I}} \ (A_5)} \end{array}$$

Fig. 3. The axiom system E_{FTP}

Axioms (A_1)–(A_4) are well-known[12]. Axiom (A_5) describes the propagation of witness constants for the case of implicit predicates.

We now introduce the notion of terms in *head normal form*. This concept plays a key role in the proofs of completeness for the axiom systems generated by our framework.

Definition 5 (Head Normal Form). *Let Σ be a signature such that $\Sigma_{FTP} \subseteq \Sigma$. A term t in $T(\Sigma)$ is in* head normal form *(for short, h.n.f.) if*

$$t = \sum_{i \in I} a_i.t_i + \sum_{j \in J} c_{P_j}, \text{ and the } P_j \text{ are all the predicates satisfied by } t.$$

The empty sum $(I = \emptyset, J = \emptyset)$ is denoted by the deadlock constant δ.

Lemma 2. *E_{FTP} is head normalizing for terms in $T(\Sigma_{FTP})$. That is, for all t in $T(\Sigma_{FTP})$, there exists t' in $T(\Sigma_{FTP})$ in h.n.f. such that $E_{FTP} \vdash t = t'$ holds.*

Proof. The reasoning is by induction on the structure of t. The whole proof is included in Appendix C. □

Theorem 1. *E_{FTP} is sound and ground-complete for bisimulation on $T(\Sigma_{FTP})$. That is, $(\forall t, t' \in T(\Sigma_{FTP}))\,.\, E_{FTP} \vdash t = t'$ iff $t \sim t$.*

Proof. The full proof is included in Appendix D. □

3.3 Axiomatizing negative premises

A crucial step in finding a complete axiomatization for *preg* systems is the "axiomatization" of negative premises (of the shape $x \overset{a}{\nrightarrow}$, $\neg Px$). In the style of [2], we introduce the initial restriction operator $\partial_{\mathcal{B},\mathcal{Q}}$, where $\mathcal{B} \subseteq \mathcal{A}$ and $\mathcal{Q} \subseteq \mathcal{P}$ are the sets of initially forbidden actions and predicates, respectively. The semantics of $\partial_{\mathcal{B},\mathcal{Q}}$ is given by the two types of transition rules in Fig. 4.

$$\frac{x \xrightarrow{a} x'}{\partial_{\mathcal{B},\mathcal{Q}}(x) \xrightarrow{a} \partial_{\emptyset,\mathcal{Q}\cap\mathcal{P}^{\mathcal{I}}}(x')} \text{ if } a \notin \mathcal{B}\ (rl_8) \qquad \frac{Px}{P(\partial_{\mathcal{B},\mathcal{Q}}(x))} \text{ if } P \notin \mathcal{Q}\ (rl_9)$$

Fig. 4. The semantics of $\partial_{\mathcal{B},\mathcal{Q}}$

Note that $\partial_{\mathcal{B},\mathcal{Q}}$ behaves like the one step restriction operator in [2] for the actions in $\mathcal{B}$, as the restriction on the action disappears after one transition. On the other hand, for the case of predicates in $\mathcal{Q}$, the operator $\partial_{\mathcal{B},\mathcal{Q}}$ resembles the CCS restriction operator [12] since, due to the presence of implicit predicates, not all the restrictions related to predicate satisfaction necessarily disappear after one step, as will become clear in what follows.

We write E^{∂}_{FTP} for the extension of E_{FTP} with the axioms involving $\partial_{\mathcal{B},\mathcal{Q}}$ presented in Fig. 6. $\mathcal{R}^{\partial}_{FTP}$ stands for the set of rules (rl_1)–(rl_9), while FTP^{∂} represents the system $(\Sigma^{\partial}_{FTP}, \mathcal{R}^{\partial}_{FTP})$.

$$\begin{array}{llll}
\partial_{\mathcal{B},\mathcal{Q}}(\delta) = \delta & (A_6) & \partial_{\mathcal{B},\mathcal{Q}}(a.x) = \sum_{P\notin\mathcal{Q},P(a.x)} c_P \text{ if } a \in \mathcal{B} & (A_9) \\
\partial_{\mathcal{B},\mathcal{Q}}(c_P) = \delta \text{ if } P \in \mathcal{Q} & (A_7) & \partial_{\mathcal{B},\mathcal{Q}}(a.x) = \partial_{\emptyset,\mathcal{Q}}(a.x) \quad \text{ if } a \notin \mathcal{B} & (A_{10}) \\
\partial_{\mathcal{B},\mathcal{Q}}(c_P) = c_P \text{ if } P \notin \mathcal{Q} & (A_8) & \partial_{\emptyset,\mathcal{Q}}(a.x) = a.\partial_{\emptyset,\mathcal{Q}\cap\mathcal{P}^{\mathcal{I}}}(x) \quad \text{ if } \mathcal{Q}\setminus\mathcal{P}^{\mathcal{I}} = \emptyset & (A_{11})
\end{array}$$
$$\partial_{\mathcal{B},\mathcal{Q}}(x+y) = \partial_{\mathcal{B},\mathcal{Q}}(x) + \partial_{\mathcal{B},\mathcal{Q}}(y)\ (A_{12})$$

Fig. 5. The axiom system $E^{\partial}_{FTP} \setminus E_{FTP}$

Axiom (A_6) states that it is useless to impose restrictions to δ, as δ does not exhibit any behaviour. The intuition behind (A_7) is that since a predicate witness c_P does not perform any action, inhibiting the satisfiability of P leads to a process with no behaviour (δ). Consequently, if the restricted predicates do not include P, the resulting process is c_P itself (see (A_8)). Inhibiting the only action a process $a.t$ can perform, leads to a new process which, in the best case, satisfies some of the predicates in $\mathcal{P}^{\mathcal{I}}$ satisfied by t (by (rl_7)) if $\mathcal{Q} \neq \mathcal{P}^{\mathcal{I}}$ (see (A_9)). Whenever the restricted action set $\mathcal{B}$ does not contain the only action a process $a.t$ can perform, then it is safe to give up $\mathcal{B}$ (see (A_{10})). As a process $a.t$ only satisfies the predicates also satisfied by t, it is straightforward that $\partial_{\emptyset,\mathcal{Q}}(a.t)$ is equivalent to the process obtained by propagating the restrictions on implicit predicates deeper into the behaviour of t (see (A_{11})). Axiom (A_{12}) is given in conformity with the semantics of $\_+\_$ ($s + t$ encapsulates both the behaviours of s and t).

Remark 1. For the sake of brevity and readability, in Fig. 4 we presented (A_9), which is a schema with infinitely many instances. However, that schema can be replaced by a finite family of axioms, as presented in Appendix E.

Theorem 2. *The following statements hold for E^{∂}_{FTP}:*

1. *E^{∂}_{FTP} is sound for bisimilarity on $T(\Sigma^{\partial}_{FTP})$.*
2. *$\forall t \in T(\Sigma^{\partial}_{FTP}), \exists t' \in T(\Sigma_{FTP})$ s.t. $E^{\partial}_{FTP} \vdash t = t'$.*

Proof. The whole proof can be found in Appendix F. □

As proving completeness for FTP^{∂} can be reduced to showing completeness for *FTP* (already proved in Theorem 1), the following result is an immediate consequence of Theorem 2:

Corollary 1. *E^{∂}_{FTP} is sound and complete for bisimulation on $T(\Sigma^{\partial}_{FTP})$.*

4 Smooth and distinctive operations

Recall that our goal consists in providing a sound and ground-complete axiomatization for bisimilarity on systems specified in the *preg* format. As the *preg* format is too permissive for achieving this result directly, our next task is to find a class of operations for which we can build such an axiomatization by "easily" reducing it to the completeness result for *FTP*, which presented in Theorem 1. In the literature, these operations are known as *smooth and distinctive* [2]. As we will see, these operations are incrementally identified by imposing suitable restrictions on *preg* rules. The standard procedure is to first find the *smooth* operations, based on which one determines the *distinctive* ones.

Definition 6 (Smooth operation).

1. *A* preg *transition rule is* smooth *if it is of the following format:*

$$\frac{\begin{array}{cc}\{x_i \xrightarrow{a_i} y_i \mid i \in I^+\} & \{P_i x_i \mid i \in J^+\} \\ \{x_i \overset{b}{\nrightarrow} \mid i \in I^-, b \in \mathcal{B}_i\} & \{\neg Q x_i \mid i \in J^-, Q \in \mathcal{Q}_i\}\end{array}}{f(x_1, \ldots, x_l) \xrightarrow{c} C[\vec{x}, \vec{y}]}$$

   where
   (a) I^+, J^+, I^-, J^- disjointly cover the set $L = \{1, \ldots, l\}$;
   (b) in the target $C[\vec{x}, \vec{y}]$ we allow only: y_i $(i \in I^+)$, x_i $(i \in I^- \cup J^-)$.
2. *A* preg *predicate rule is* smooth *if it has the form above, its premises satisfy condition (1a) and its conclusion is $P(f(x_1, \ldots, x_l))$ for some $P \in \mathcal{P}$.*
3. *An operation f of a* preg *system is* smooth *if all its (transition and predicate) rules are smooth.*

By Definition 6, a rule ρ is smooth if it satisfies the following properties:

– a position i cannot be tested both positively and negatively at the same time,
– positions tested positively are not tested for the performance of multiple transitions (respectively, for the satisfiability of multiple predicates) within the same rule, and
– if ρ is a transition rule, then the occurrence of variables at positions $i \in I^+ \cup J^+$ is not allowed in the target.

Remark 2. Note that we can always consider a position i that does not occur as a premise in a rule for f as being negative, with the empty set of constraints (i.e. either $i \in I^-$ and $\mathcal{B}_i = \emptyset$, or $i \in J^-$ and $\mathcal{Q}_i = \emptyset$).

Definition 7 (Distinctive operation). *An operation f of a* preg *system is* distinctive *if:*

– *for each argument i, either all rules for f test i positively, or none of them does, and*
– *for any two distinct rules for f there exists a position i tested positively, such that one of the following holds:*
  - *both rules have actions that are different in the premise at position i;*
  - *both rules have predicates that are different in the premise at position i;*
  - *one rule has an action premise at position i, and the other rule has a predicate test at the same position i.*

According to the first requirement in Definition 7, we state that for a smooth and distinctive operation f, a position i is *positive* (respectively, *negative*) for f if there is a rule for f such that i is tested positively (respectively, negatively) for that rule.

The existence of a family of smooth and distinctive operations "describing the behaviour" of a general *preg* operation is formalized by the following lemma:

Lemma 3. *Consider a* preg *system G. Then there exist a* preg *system G', which is a disjoint extension of G and FTP, and a finite axiom system E such that*

1. *E is sound for bisimilarity over any disjoint extension G'' of G', and*
2. *for each term t in $T(\Sigma_G)$ there is some term t' in $T(\Sigma_{G'})$ such that t' is built solely using smooth and distinctive operations and E proves $t = t'$.*

A detailed description of the transformation process from general *preg* to smooth and distinctive operations is provided in Appendix G.

4.1 Axiomatizing smooth and distinctive *epreg* operations

We will now show how to axiomatize smooth and distinctive operations.

From this point forward, for ease of presentation, we will use only explicit predicates (*i.e.*, we take $\mathcal{P}^{\mathcal{I}} = \emptyset$) in our theory. The theory that we present in what follows can be extended to deal with implicit predicates, at the cost of imposing some "sanity conditions" on the *preg* specifications of predicates in $\mathcal{P}^{\mathcal{I}}$.

We hint at the constraints on *preg* languages that make this extension possible in Appendix H. We define *epreg* – the restriction of *preg* to explicit predicates. Note that all the results so far remain valid also for the *epreg* case. Rules of type (rl_7) are not a part of our predicate specification framework anymore. Consequently, (A_5) is never applied and the restriction operator $\partial_{\mathcal{B},\mathcal{Q}}$ disappears after an action performance, as there is no need to propagate constraints dealing with implicit predicates.

Note that the restriction to *epreg* systems is not necessarily a limitation since one can always explicitly consider enough predicate witnesses to express implicit predicates satisfiability. Consider the case of the eventual termination predicate $\lightning$. Checking for the eventual termination of $a.a.c_{\lightning}$ is reduced to verifying the explicit termination of $a.(a.c_{\lightning} + c_{\lightning}) + c_{\lightning}$. Moreover, our tool has the capability to carry out this transformation automatically.

In what follows we proceed in a similar fashion to [2]. We define a set of laws used in the construction of a complete axiomatization for bisimilarity on terms built over smooth and distinctive operations. The strength of these laws lies in their capability of reducing terms to their head normal form, thus reducing completeness for general *epreg* systems to completeness of E_{FTP} (which was already proved in Section 3.2).

Definition 8. *Let f be a smooth and distinctive l-ary operation of an* epreg *system G, such that $FTP^{\partial} \sqsubseteq G$.*

1. *For a positive position $i \in L = \{1, \ldots, l\}$, the* distributive law *for i w.r.t. f is given as follows:*

$$f(X_1, \ldots, X_i' + X_i'', \ldots, X_l) = f(X_1, \ldots, X_i', \ldots, X_l) + f(X_1, \ldots, X_i'', \ldots, X_l)$$

2. *For a rule $\rho \in \mathcal{R}$ for f the* trigger law *is, depending on whether ρ is a transition or a predicate rule:*

$$f(\vec{X}) = \begin{cases} c.C[\vec{X}, \vec{y}]\ ,\ \rho \in \mathcal{R}^{\mathcal{A}} \text{ (action law)} \\ c_P\ ,\ \rho \in \mathcal{R}^{\mathcal{P}} \text{ (predicate law)} \end{cases}$$

where

$$X_i \equiv \begin{cases} a_i.y_i\ ,\ i \in I^+ \\ c_{P_i}\ ,\ i \in J^+ \\ \partial_{\mathcal{B}_i,\mathcal{Q}_i}(x_i)\ ,\ i \in I^- \cup J^- \end{cases}$$

3. *Suppose that for $i \in L$, term X_i is of the form $\delta, c_{P_i}, a.z_i, a.z_i + z_i'$, $c_{P_i} + z_i$ or z_i. Suppose further that for each rule for f there exists $X_j \in \vec{X}$ ($j \in \{1, \ldots, l\}$) s.t. one of the following holds:*
   - *$j \in I^+$ and ($X_j \equiv \delta$ or $X_j \equiv a.z_j$ ($a \neq a_j$) or $X_j \equiv c_P$, for some P),*
   - *$j \in J^+$ and ($X_j \equiv \delta$ or $X_j \equiv c_P$ ($P \neq P_j$) or $X_j \equiv a_j.z_j$, for some a_j),*
   - *$j \in I^-$ and $X_j \equiv b.z_j + z_j'$, where $b \in \mathcal{B}_j$,*
   - *$j \in J^-$ and $X_j \equiv c_Q + z_j$, where $Q \in \mathcal{Q}_j$.*

Then the deadlock law *is as follows:*

$$f(\vec{X}) = \delta.$$

Theorem 3. *Consider G an* epreg *system such that $FTP^{\partial} \sqsubseteq G$. Let $\Sigma \subseteq \Sigma_G \setminus \Sigma^{\partial}_{FTP}$ be a collection of smooth and distinctive operations of G. Let E_G be the finite axiom system that extends E^{∂}_{FTP} with the following axioms for each $f \in \Sigma$:*

- *for each positive argument i of f, a distributivity law (Definition 8.1),*
- *for each transition rule for f, an action law (Definition 8.2),*
- *for each predicate rule for f, a predicate law (Definition 8.2), and*
- *all deadlock laws for f (Definition 8.3).*

The following statements hold for E_G, for any G' such that $G \sqsubseteq G'$:

1. *E_G is sound for bisimilarity on $T(\Sigma \cup \Sigma^{\partial}_{FTP})$.*
2. *E_G is head normalizing for $T(\Sigma \cup \Sigma^{\partial}_{FTP})$.*

Proof. The full proof is presented in Appendix I. □

5 Soundness and completeness

Let us summarize our results so far. By Theorem 3, it follows that, for any *epreg* system $G \sqsupseteq FTP^{\partial}$, there is an axiomatization that is head normalizing for $T(\Sigma \cup \Sigma^{\partial}_{FTP})$, where $\Sigma \subseteq \Sigma_G \setminus \Sigma^{\partial}_{FTP}$ is a collection of smooth and distinctive operations of G. Also, as hinted in Section 4 (Lemma 3), there exists a sound algorithm for transforming general *preg* operations to smooth and distinctive ones.

So, for any *epreg* system G, we can build an *epreg* system $G' \sqsupseteq G$ and an axiomatization $E_{G'}$ that is head normalizing for $T(\Sigma_{G'})$. This statement is formalized as follows:

Theorem 4. *Let G be an* epreg *system. Then there exist $G' \sqsupseteq G$ and a finite axiom system $E_{G'}$ such that*

1. *$E_{G'}$ is sound for bisimulation on $T(\Sigma_{G'})$,*
2. *$E_{G'}$ is head normalizing for $T(\Sigma_{G'})$,*

and moreover, G' and $E_{G'}$ can be effectively constructed from G.

Proof. The proof follows immediately by Theorem 3 and by the existence of an algorithm used for transforming general *preg* to smooth and distinctive operations, described in Appendix G. □

Remark 3. Theorem 4 guarantees ground-completeness of the generated axiomatization for well-founded *epreg* specifications, that is, *epreg* specifications in which each process can only exhibit finite behaviour.

Let us further recall an example given in [2]. Consider the constant ω, specified by the rule $\omega \xrightarrow{a} \omega$. Obviously, the corresponding action law $\omega = a.\omega$ will apply for an infinite number of times in the head normalization process. So the last step in obtaining a complete axiomatization is to handle infinite behaviour.

Let t and t' be two processes with infinite behaviour (remark that the infinite behaviour is a consequence of performing actions for an infinite number of times, so the extension to predicates is not a cause for this issue). Since we are dealing with finitely branching processes, it is well known that two process terms are bisimilar for each finite depth n. One way of formalizing this requirement is to use the well-known *Approximation Induction Principle* (AIP) [5, 6].

Let us first consider the operations $\pi_n(\cdot)$, $n \in \mathbb{N}$, known as *projection operations.* The purpose of these operations is to stop the evolution of processes after a certain number of steps. The AIP is given by the following conditional equation:

$$x = y \text{ if } \pi_n(x) = \pi_n(y) \ (\forall n \in \mathbb{N}).$$

We further adapt the idea in [2] to our context, and model the infinite family of projection operations $\pi_n(\cdot)$, $n \in \mathbb{N}$, by a binary operation $\cdot/\cdot$ defined as follows:

$$\frac{x \xrightarrow{a} x' \quad h \xrightarrow{c} h'}{x/h \xrightarrow{a} x'/h'}\ (rl_{10}) \qquad \frac{Px}{P(x/h)}\ (rl_{11})$$

where c is an arbitrary action. Note that $\cdot/\cdot$ is a smooth and distinctive operation.

The role of variable h is to "control" the evolution of a process, *i.e.*, to stop the process in performing actions, after a given number of steps. Variable h (the "hourglass" in [2]) will always be instantiated with terms of the shape c^n, inductively defined as: $c^0 = \delta$, $c^{n+1} = c.c^n$.

Let $G = (\Sigma_G, \mathcal{R}_G)$ be an *epreg* system. We use the notation $G_/$ to refer to the *epreg* system $(\Sigma_G \cup \{\cdot/\cdot\}, \mathcal{R}_G \cup \{(rl_{10}), (rl_{11})\})$ – the extension of G with $\cdot/\cdot$. Moreover, we use the notation E_{AIP} to refer the axioms for the smooth and distinctive operation $\cdot/\cdot$, derived as in Section 4.1 – Definition 8.

We reformulate AIP according to the new operation $\cdot/\cdot$:

$$x = y \text{ if } x/c^n = y/c^n \ (\forall n \in \mathbb{N})$$

Lemma 4. *AIP is sound for bisimulation on* $T(\Sigma_{FTP_/})$.

Proof. The whole proof can be found in Appendix J. □

In what follows we provide the final ingredients for proving the existence of a ground-complete axiomatization for bisimilarity on *epreg* systems. As previously stated, this is achieved by reducing completeness to proving equality in *FTP*. So, based on AIP, it would suffice to show that for any closed process term t and natural number n, there exists an *FTP* term equivalent to t at moment n in time:

Lemma 5. *Consider G an* epreg *system. Then there exist $G' \sqsupseteq G_/$ and $E_{G'}$ with the property:* $\forall t \in T(\Sigma_{G'}), \forall n \in \mathbb{N}, \exists t' \in T(\Sigma_{FTP})$ *s.t.* $E_{G'} \vdash t/c^n = t'$.

At this point we can prove the existence of a sound and ground-complete axiomatization for bisimulation equivalence on general *epreg* systems:

Theorem 5 (Soundness and Completeness). *Consider G an* epreg *system. Then there exist $G' \sqsupseteq G_{/}$ and $E_{G'}$ a finite axiom system, such that $E_{G'} \cup E_{AIP}$ is sound and complete for bisimulation on $T(\Sigma_{G'})$.*

6 Conclusions and future work

In this paper we have introduced the *preg* rule format, a natural extension of GSOS with arbitrary predicates. Moreover, we have provided a procedure (similar to the one in [2]) for deriving sound and ground-complete axiomatizations for bisimilarity of systems that match the *epreg* format (that is, *preg* restricted to explicit predicates). In the current approach, explicit predicates are handled by considering constants witnessing their satisfiability as summands in tree expressions. Consequently, there is no explicit predicate P satisfied by a term of shape $\Sigma_{i \in I} a_i.t_i$.

The procedure introduced in this paper has also enabled the implementation of a tool that can be used to automatically reason on bisimilarity of systems specified as terms built over operations defined by *epreg* rules.

Several possible extensions are left as future work. As a first attempt we consider investigating further ways of handling implicit predicates (given by (rl_7)). One way to achieve this is to stipulate certain restrictions (consistency requirements) on the language specifications, as indicated in Appendix H.

As our framework currently allows only for the specification of existential predicates (given by (rl_5) and (rl_6)), another possible extension would be the explicit handling of universal predicates over trees (given by rules of the form $\frac{Px \ Py}{P(x+y)}$). Currently, one may indirectly handle these predicates by considering their negation, which is an existential predicate (*e.g.* instead of using *convergence*, one needs to be content with *divergence*).

It would also be worth investigating the properties of positive *preg* languages. By allowing only positive premises we eliminate the need of the restriction operators ($\partial_{\mathcal{B},\mathcal{Q}}$) during the axiomatization process. This would enable us to deal with more general predicates over trees, such as those that may be satisfied by terms of the form $a.t$ where a ranges over some subset of the collection of actions.

Acknowledgments. The authors are grateful for the useful comments and suggestions from Alexandra Silva.

References

1. L. Aceto. Deriving complete inference systems for a class of GSOS languages generation regular behaviours. In Jonsson and Parrow [11], pages 449–464.
2. L. Aceto, B. Bloom, and F. Vaandrager. Turning SOS rules into equations. *Inf. Comput.*, 111:1–52, May 1994.

3. J. C. M. Baeten, T. Basten, and M. A. Reniers. *Process Algebra: Equational Theories of Communicating Processes*, volume 50 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, Cambridge, 2010.
4. J. C. M. Baeten and E. P. de Vink. Axiomatizing GSOS with termination. *J. Log. Algebr. Program.*, 60-61:323–351, 2004.
5. J. C. M. Baeten and W. P. Weijland. *Process algebra*. Cambridge University Press, New York, NY, USA, 1990.
6. J. A. Bergstra and J. W. Klop. Verification of an alternating bit protocol by means of process algebra. In *Proceedings of the International Spring School on Mathematical method of specification and synthesis of software systems '85*, pages 9–23, New York, NY, USA, 1986. Springer-Verlag New York, Inc.
7. B. Bloom, S. Istrail, and A. R. Meyer. Bisimulation can't be traced. *J. ACM*, 42:232–268, January 1995.
8. M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer, and C. L. Talcott, editors. *All About Maude - A High-Performance Logical Framework, How to Specify, Program and Verify Systems in Rewriting Logic*, volume 4350 of *Lecture Notes in Computer Science*. Springer, 2007.
9. M. Gazda and W. Fokkink. Turning GSOS into equations for linear time-branching time semantics. *2nd Young Researchers Workshop on Concurrency Theory - YR-CONCUR'10, Paris*, 2010.
10. C. Hoare. *Communicating Sequential Processes*. Prentice-Hall International, Englewood Cliffs, 1985.
11. B. Jonsson and J. Parrow, editors. *CONCUR '94, Concurrency Theory, 5th International Conference, Uppsala, Sweden, August 22-25, 1994, Proceedings*, volume 836 of *Lecture Notes in Computer Science*. Springer, 1994.
12. R. Milner. *Communication and Concurrency*. Prentice-Hall International, Englewood Cliffs, 1989.
13. D. Park. Concurrency and automata on infinite sequences. In P. Deussen, editor, *5th GI Conference,* Karlsruhe, Germany, volume 104 of *Lecture Notes in Computer Science*, pages 167–183. Springer-Verlag, 1981.
14. I. Ulidowski. Finite axiom systems for testing preorder and de simone process languages. *Theor. Comput. Sci.*, 239(1):97–139, 2000.
15. R. van Glabbeek. The linear time - branching time spectrum I. The semantics of concrete, sequential processes. In A. P. S. S. J. Bergstra, editor, *Handbook of Process Algebra*, pages 3–99. Elsevier, 2001.

A Discussion on predicate rules for trees

In what follows we provide the reader with a brief description of a) the capabilities of the framework used for predicate specification; b) the reasons for the design decisions we have made developing the framework defined above.

First, note that our framework allows for the definition of *existential predicates* $P \in \mathcal{P}$ by using rules of the form $\left\{ \dfrac{Px}{P(x+y)}(rl_5), \dfrac{Py}{P(x+y)}(rl_6) \right\}$. We refer to these predicates as existential because in order for a sum to satisfy a predicate it is sufficient to find a summand that satisfies it. Since closed terms in the language FTP essentially describe finite synchronization trees, this means that a tree satisfies a predicate if so does one of its branches. "Orthogonally", we allow for (but do not impose) specifying *implicit predicates* using rules of the form $\left\{ \dfrac{Px}{P(a.x)}(rl_7) \right\}$. Moreover, remark that there is no way to infer $P\delta$, so we always consider that $\neg P\delta$ holds for each predicate P. Basically, δ has no transitions and does not satisfy any predicate. A constant c_P, in comparison, has no transitions, but it does satisfy the predicate P, and only that predicate.

As a first attempt, we also wanted our framework to allow for rules of the shape $\left\{ \dfrac{}{P(x+y)} \right\}$ or $\left\{ \dfrac{Px}{P(x+y)} \right\}$. But we decided to eliminate them, as they would invalidate standard algebraic properties such as the idempotence and the commutativity of $\_+\_$.

Without loss of generality we avoid rules of the form $\left\{ \dfrac{}{P(a.x)} \right\}$. As far as the user is concerned, in order to express that $a.x$ satisfies a predicate P, one can always add the witness c_P as a summand: $a.x + c_P$. This decision helped us avoid some technical problems for the soundness and completeness proofs for the case of the restriction operator $\partial_{\mathcal{B},\mathcal{Q}}$, which is presented in Section 3.3.

Due to the aforementioned restriction, we also had to leave out universal predicates given by $\mathcal{P}^{\forall} = \left\{ \dfrac{Px \; Py}{P(x+y)} \right\}$. Note that the elimination of universal predicates is not a theoretical limitation to what one can express, since a universal predicate can always be defined as the negation of an existential one.

In what follows we introduce some ideas enabling the integration of universal predicates as first-class notions within our framework. Intuitively, the property $P \in \mathcal{P}^{\forall}$ holds for a "tree" t (written Pt) if and only if it holds for each of the branches of t. It is reasonable, therefore, to consider that $P\delta$ holds. This is actually mandatory in order to preserve the validity of the axiom $x + \delta = x$. In this context, we are also interested in how universal predicates interact with action prefixing. It is safe to assume that for each universal predicate P, there is an axiom of the form $(U)\dfrac{}{P(a.x)}$ for at least one action a (otherwise we would be in the trivial case in which P is only satisfied by δ).

In order to implement universal predicates we consider a constant d_P for each $P \in \mathcal{P}^{\forall}$. d_P has no transitions and does *not* satisfy the predicate P, but satisfies any other universal predicate (this constant can be regarded as a *negative witness* for P). Moreover, we consider that no universal predicate is implicit, so $\mathcal{P}^{\mathcal{I}} \subseteq \mathcal{P}^{\exists}$ (the set of existential predicates) – this is reasonable because of (U).

As a last approach, we thought of allowing the user to specify existential predicates using rules of the form $\dfrac{P_1 x \ldots P_n x}{P(x+y)}(*)$ and $\dfrac{P_1 y \ldots P_n y}{P(x+y)}(**)$ (instead of (rl_5) and (rl_6)). However, in order to maintain the validity of the axiom $x + x = x$ in the presence of rules of these forms, it would have to be the case that one of the predicates P_i in the premises is P itself. [If that were not the case, then let t be the sum of the constants witnessing the P_i's for a rule of the form $(*)$ above with a minimal set of set premises. We have that $t + t$ satisfies P by rule $(*)$. On the other hand, Pt does not hold since none of the P_i is equal to P and no rule for P with a smaller set of premises exists.] Now, if a rule of the form $(*)$ has a premise of the form Px, then it is subsumed by (rl_5) which we must have to ensure the validity of laws such as $c_P = c_P + c_P$.

B Sound and supported relations

Definition 9 (Transition relation). *A* transition relation *over a signature Σ is a relation $\rightsquigarrow \subseteq T(\Sigma) \times \mathcal{A} \times T(\Sigma)$. We write $t \overset{a}{\rightsquigarrow} t'$ as an abbreviation for $(t, a, t') \in \rightsquigarrow$.*

Definition 10 (Predicate relation). *A* predicate relation *over a signature Σ is a relation $\propto \subseteq \mathcal{P} \times T(\Sigma)$. We write Pt as an abbreviation for $(P, t) \in \propto$.*

Definition 11 (Σ-substitution). *A* (closed) Σ-substitution *is a function σ from variables to (closed) terms over the signature Σ. For a term S, we write $S\sigma$ for the result of substituting $\sigma(x)$ for each x in S. For $\vec{S}$ and $\vec{x}$ of the same length, $\langle \vec{S}/\vec{x} \rangle$ represents the substitution that replaces the i-th variable of $\vec{x}$ by the i-th term of $\vec{S}$, and is the identity function on all the other variables.*

Definition 12 (Rule satisfiability). *Consider $\rightsquigarrow$ a transition relation, $\propto$ a predicate relation, and σ a closed substitution. For each (transition, resp. predicate) formula γ, the predicate $\rightsquigarrow, \propto, \sigma \models \gamma$ is defined as follows:*

$$\begin{aligned}
\rightsquigarrow, \propto, \sigma &\models S \xrightarrow{a} S' &&\triangleq (S\sigma, a, S'\sigma) \in \rightsquigarrow, \\
\rightsquigarrow, \propto, \sigma &\models S \overset{a}{\nrightarrow} &&\triangleq \nexists t \;.\; (S\sigma, a, t) \in \rightsquigarrow, \\
\rightsquigarrow, \propto, \sigma &\models PS &&\triangleq (P, S\sigma) \in \propto, \\
\rightsquigarrow, \propto, \sigma &\models \neg PS &&\triangleq (P, S\sigma) \notin \propto.
\end{aligned}$$

Let H be a set of (transition and/or predicate) formulas. We define

$$\rightsquigarrow, \propto, \sigma \models H \triangleq (\forall \gamma \in H) \;.\; \rightsquigarrow, \propto, \sigma \models \gamma$$

Let $\rho = \dfrac{H}{\gamma}$ be a preg *rule. We define*

$$\rightsquigarrow, \propto, \sigma \models \rho \triangleq (\rightsquigarrow, \propto, \sigma \models H) \Rightarrow (\rightsquigarrow, \propto, \sigma \models \gamma)$$

Definition 13 (Sound and supported relations). *Consider $G = (\Sigma_G, \mathcal{R}_G)$ a* preg *system, $\rightsquigarrow$ a transition relation over Σ_G, and $\propto$ a predicate relation over Σ_G. Then $\rightsquigarrow$ and $\propto$ are* sound *for G iff for every rule $\rho \in \mathcal{R}_G$ and every closed substitution σ, we have $\rightsquigarrow, \propto, \sigma \models \rho$.*
A transition formula $t \overset{a}{\rightsquigarrow} t'$ (resp. a predicate formula Pt) is supported *by some rule $\rho = \dfrac{H}{\gamma} \in \mathcal{R}_G$ iff there exists a substitution σ s.t. $\rightsquigarrow, \propto, \sigma \models H$ and $\gamma\sigma = t \xrightarrow{a} t'$ (resp. $\gamma\sigma = Pt$). Relations $\rightsquigarrow$ and $\propto$ are* supported *by G iff each of their (transition, resp. predicate) formulas are supported by a rule in $\mathcal{R}_G$.*

Lemma 6. *For each* preg *system G there exists a unique sound and supported transition relation, and a unique sound and supported predicate relation.*

Proof. We start by showing that $\forall t \in T(\Sigma_G)$, the following sets exist and are uniquely defined:

$$\rightsquigarrow (t) = \{(a, t') \mid t \xrightarrow{a} t',\, a \in \mathcal{A}\} \text{ and } \propto(t) = \{P \mid Pt,\, P \in \mathcal{P}\}$$

Let $t = f(t_1, \ldots, t_n) \in T(\Sigma_G)$. In order to determine $\rightsquigarrow (t)$ and $\propto(t)$, we exploit the properties:

1. $(a, t') \in \rightsquigarrow (t)$ iff $\exists R = \dfrac{H}{f(\vec{x}) \xrightarrow{a} C[\vec{x}, \vec{y}]} \in \mathcal{R}^{\mathcal{A}}$ and σ a substitution s.t.:
   (a) $\sigma(x_i) = t_i$ $(\forall i \in \{1, \ldots, n\})$
   (b) $C[\vec{x}, \vec{y}]\sigma = t'$
   (c) $\forall x_i \xrightarrow{a_{ij}} y_{ij} \in H$, it holds that $(a_{ij}, \sigma(y_{ij})) \in \rightsquigarrow (t_i)$
   (d) $\forall x_i \overset{b}{\nrightarrow} \in H$, it holds that $\nexists t'_i.(b, t'_i) \in \rightsquigarrow (t_i)$
   (e) $\forall P_{ij} x_i \in H$, it holds that $P_{ij} \in \propto(t_i)$
   (f) $\forall \neg P_{ij} x_i \in H$, it holds that $P_{ij} \notin \propto(t_i)$
2. $P \in \propto(t)$ iff $\exists R = \dfrac{H}{P(f(\vec{x}))} \in \mathcal{R}^{\mathcal{P}}$ and σ a substitution s.t.:
   (a) $\sigma(x_i) = t_i$ $(\forall i \in \{1, \ldots, n\})$
   (b) $\forall x_i \xrightarrow{a_{ij}} y_{ij} \in H$, it holds that $(a_{ij}, \sigma(y_{ij})) \in \rightsquigarrow (t_i)$
   (c) $\forall x_i \overset{b}{\nrightarrow} \in H$, it holds that $\nexists t'_i.(b, t'_i) \in \rightsquigarrow (t_i)$
   (d) $\forall P_{ij} x_i \in H$, it holds that $P_{ij} \in \propto(t_i)$
   (e) $\forall \neg P_{ij} x_i \in H$, it holds that $P_{ij} \notin \propto(t_i)$

We further determine $\rightsquigarrow (t)$ and $\propto(t)$ by induction on the structure of t.

– *Base case:* f is a constant symbol. For this case, conditions (1a) and (1c)–(1f) are all satisfied. Moreover, there are no occurrences of variables in the target, so the target has to be a closed term C. Therefore, we take $t' = C$ and consider (a, t') an element of the relation $\rightsquigarrow (t)$. Similarly for the predicate rules – $P \in \propto(t)$.

– *Induction step:* $t = f(t_1, \ldots, t_n)$. By the inductive hypothesis, the sets $\rightsquigarrow (t_i)$ and $\propto(t_i)$ exist ($\forall i \in \{1, \ldots, n\}$). So, identifying all the substitutions σ that satisfy the conditions in 1. and 2. would suffice to determine the elements in $\rightsquigarrow (t)$ and $\propto(t)$, respectively.

According to the reasoning above, it is obvious that $\rightsquigarrow (t)$ and $\propto(t)$ are unique. Now take $\rightarrow_G = \{(t, a, t') \mid t \in T(\Sigma_G),\ a \in \mathcal{A} \text{ and } (a, t') \in \rightsquigarrow (t)\}$ and $\ltimes_G = \cup_{t \in T(\Sigma_G)} \propto(t) \times \{t\}$. It follows immediately that, by construction, both $\rightarrow_G$ and $\ltimes_G$ are unique, sound and supported. □

C Proof of Lemma 2

Proof. The reasoning is by induction on the structure of t.

Base cases
– $t = \delta \Longrightarrow t' = \delta$ (h.n.f) $\Longrightarrow E_{FTP} \vdash t = t'$ (by reflexivity)
– $t = c_P \Longrightarrow t' = c_P$ (h.n.f.) $\Longrightarrow E_{FTP} \vdash t = t'$ (by reflexivity)

Inductive step cases
– $t = a.t'$; t' has a simpler structure than t, so, by the inductive hypothesis, $\exists t'_1$ in h.n.f. s.t. $E_{FTP} \vdash t' = t'_1$. As $\vdash$ is a congruence, $E_{FTP} \vdash a.t' = a.t'_1$, so $E_{FTP} \vdash t = a.t'_1$. Now, $P(t)$ holds for some predicate P iff $P(t')$ and $P \in \mathcal{P}^{\mathcal{I}}$. Since t'_1 is the h.n.f. for t', it has the summand c_P for each P s.t. $P(t')$. By using the instance of (A_5) for those predicates P satisfied by t' for which there is a rule of form (rl_7) for P, we add a c_P summand to the h.n.f. for t. We obtain, therefore, $E_{FTP} \vdash t = a.t'_1 + \sum_{j \in J} c_{P_j}$ which is in h.n.f., where $\{P_j \mid j \in J\}$ is the set of predicates in $\mathcal{P}^{\mathcal{I}}$ that are satisfied by t'.
– $t = t_1 + t_2$, so $\exists t'_1, t'_2$ in h.n.f. s.t. $E_{FTP} \vdash t_i = t'_i, i \in \{1, 2\}$. By the congruence of $\vdash$ we have $E_{FTP} \vdash t_1 + t_2 = t'_1 + t'_2$. Using $(A_1), (A_2)$, we infer $E_{FTP} \vdash t_1 + t_2 = t'_{12}$, which is in h.n.f. (we could detaliate by splitting $t'_{1,2}$ into "action summators" and "witness sumators").

□

D Proof of Theorem 1

We first introduce a lemma that will be used in our proof:

Lemma 7. *Let G be a* preg *system. If*

– $t_1, t_2 \in T(\Sigma_G)$ *s.t.* $t_1 \xrightarrow{a} t$ *iff* $t_2 \xrightarrow{a} t$ *for all* $a \in \mathcal{A}$ *and for all* $t \in T(\Sigma_G)$
– Pt_1 *iff* Pt_2 *for any* $P \in \mathcal{P}$

then $t_1 \sim t_2$.

Proof. Follows directly from Definition 3 and the reflexivity of $\sim$. □

We further continue with the proof of Theorem 1.

Proof.

The soundness ($E_{FTP} \vdash s = t \Longrightarrow s \sim t$) follows in a standard fashion.

$\underline{\text{Proving } s \sim t \Longrightarrow E_{FTP} \vdash s = t \text{ [completeness]}}$

Let us first define the function *height* that computes the height of the syntactic tree associated to a term $t \in T(\Sigma_{FTP})$:

$$\mathit{height}(t) = \begin{cases} 0 & \text{if } t \in \{\delta\} \cup \{c_P \mid P \in \mathcal{P}\} \\ 1 + \max(\mathit{height}(t_1), \mathit{height}(t_2)) & \text{if } t = t_1 + t_2 \\ 1 + \mathit{height}(t') & \text{if } t = a.t' \end{cases}$$

Note that we may assume (by Lemma 2) that s, t are in head normal form.
We prove the property by induction on $M = \max(\mathit{height}(s), \mathit{height}(t))$.

Base case

– $M = 0 \Longrightarrow \begin{cases} \text{either } s = t = \delta, \text{ so } E_{FTP} \vdash s = t \\ \quad\text{or } s = t = c_P, \text{ so } E_{FTP} \vdash s = t \end{cases}$

Inductive step case($M > 0$) We show that $t = s + t$. In order to do that we argue that each summand of s is provably equal to a summand of t.

– let $a.s'$ be a summand of $s \Longrightarrow s \xrightarrow{a} s'$. As $s \sim t$, it follows that $t \xrightarrow{a} t'$ for some t' and $s' \sim t'$. Now $\max(\mathit{height}(s'), \mathit{height}(t')) < M \overset{i.h.}{\Longrightarrow} E_{FTP} \vdash s' = t'$, hence $E_{FTP} \vdash a.s' = a.t'$;
– let c_P be a summand of s (Ps holds). As $s \sim t$, Pt. t is in h.n.f., therefore c_P is also a summand of t.

So, every summand of s can be proved equal to a summand of t, so $E_{FTP} \vdash t = s + t$. (*)

By symmetry it follows that $s = t + s$ (**).

By (*,**), the congruence of $\vdash$ and ($A_1 - A_3$), we have $E_{FTP} \vdash s = t$. □

Notice that we do not explicitly use (A_5) when proving the completeness of the axiom system E_{FTP}. This is because we work with processes in h.n.f.

E Axiom (A_9), a schema with infinitely many instances

In what follows we provide a finite family of axioms that can replace (A_9):

$$\partial_{\mathcal{B},\mathcal{Q}}(a.\delta) = \delta \text{ if } a \in \mathcal{B} \qquad (A_{9.1})$$

$$\partial_{\mathcal{B},\mathcal{Q}}(a.c_P) = \begin{cases} c_P \text{ if } a \in \mathcal{B} \text{ and } (P \in \mathcal{P}^{\mathcal{I}} \text{ and } P \notin \mathcal{Q}) \\ \delta \text{ if } a \in \mathcal{B} \text{ and } (P \notin \mathcal{P}^{\mathcal{I}} \text{ or } P \in \mathcal{Q}) \end{cases} \qquad (A_{9.2})$$

$$\partial_{\mathcal{B},\mathcal{Q}}(a.b.x) = \partial_{\mathcal{A},\mathcal{Q}}(x) \text{ if } a \in \mathcal{B} \qquad (A_{9.3})$$

$$\partial_{\mathcal{B},\mathcal{Q}}(a.(x+y)) = \partial_{\mathcal{B},\mathcal{Q}}(a.x) + \partial_{\mathcal{B},\mathcal{Q}}(a.y) \text{ if } a \in \mathcal{B} \qquad (A_{9.4})$$

Fig. 6. Axioms replacing (A_9)

Note that replacing (A_9) with $(A_{9.1})$ – $(A_{9.4})$ does not affect the validity of the statements in Theorem 2.

Soundness (statement 1) follows in a standard fashion, similar to Theorem 1.

In order to prove statement 2 we proceed as follows. Assume t is a tree term and $a \in \mathcal{B}$. We prove that there is a h.n.f t' s.t. $E^{\partial}_{FTP} \vdash \partial_{\mathcal{B},\mathcal{Q}}(a.t) = t'$, using $(A_{9.1})$ – $(A_{9.4})$. The result follows by induction on the structure of t:

- $t = \delta$. Then use $(A_{9.1})$.
- $t = c_P$. Then use $(A_{9.2})$.
- $t = b.t''$. Use $(A_{9.3})$ and induction.
- $t = t_1 + t_2$. Use $(A_{9.4})$ and induction.

F Proof of Theorem 2

Proof.

Proving $E^{\partial}_{FTP} \vdash s = t \Longrightarrow s \sim t$ [soundness]

The property follows directly for $(A_{6,7,8})$ because neither the lhs, nor the rhs terms can "advance" with actions and their (in)satisfiability of predicates is easily checkable.

Take (A_9). Let $s = \partial_{\mathcal{B},\mathcal{Q}}(a.\overline{s})$, with $a \in \mathcal{B}$. Prove that $s \sim t = \sum_{P \notin \mathcal{Q}, P(a.\overline{s})} c_P$.

- As $a \in \mathcal{B}$ it follows that $s \overset{a}{\nrightarrow}$. Also $s \overset{b}{\nrightarrow}$ for any $b \in Act, b \neq a$. As t is a sum of constants, $t \overset{b}{\nrightarrow}$ for any $t \in Act$.
- If $Ps \overset{rl_9}{\Longrightarrow} P \notin \mathcal{Q}$ and $P(a.\overline{s})$. Then Pt also holds because c_P is one of its summands. For the other direction, if Pt, then $P(a.\overline{s})$ and $P \notin \mathcal{Q} \overset{rl_9}{\Longrightarrow} P(\partial_{\mathcal{B},\mathcal{Q}}(a.\overline{s}))$.

Therefore, by Lemma 7, it follows that $s \sim t$.

Take (A_{10}). Let $s = \partial_{\mathcal{B},\mathcal{Q}}(a.\overline{s})$ and $a \notin \mathcal{B}$. Prove that $s \sim t = \partial_{\emptyset,\mathcal{Q}}(a.\overline{s})$.

– If $s \xrightarrow{b} s'$ for some fixed $b \in Act$ and $s' \in S$, then by (rl_8), it follows that $b \notin \mathcal{B}$, $s' = \partial_{\emptyset,\mathcal{Q}}(s'')$, and $a.\overline{s} \xrightarrow{b} s''$. It follows $b = a$, $s'' = \overline{s}$, and, therefore $s = \partial_{\mathcal{B},\mathcal{Q}}(a.\overline{s}) \xrightarrow{a} \partial_{\emptyset,\mathcal{Q}}(\overline{s})$. Now, does $t = \partial_{\emptyset,\mathcal{Q}}(a.\overline{s}) \xrightarrow{a} \partial_{\emptyset,\mathcal{Q}}(\overline{s})$ hold ? Yes, it does, because $a \notin \emptyset$ and (rl_8) can be used. Similarly for $t \xrightarrow{b} t'$.
– If $P(s = \partial_{\mathcal{B},\mathcal{Q}}(a.\overline{s}))$, then $\overset{(rl_9)}{\Longrightarrow} P \notin \mathcal{Q}$ and $P(a.\overline{s}) \overset{(rl_9)}{\Longrightarrow} P(\partial_{\emptyset,\mathcal{Q}}(a.\overline{s}) = t)$. Similarly for Pt.

Therefore, by Lemma 7, it follows that $s \sim t$.

Take (A_{11}). Let $s = \partial_{\emptyset,\mathcal{Q}}(a.\overline{s})$. Prove that $s \sim t = a.\partial_{\emptyset,\mathcal{Q}\cap\mathcal{P}^{\mathcal{I}}}(\overline{s})$.

– If $s \xrightarrow{b} s'$ for some fixed $b \in Act$ and $s' \in S$, then by (rl_8), it follows that $b \notin \mathcal{B}$, $s' = \partial_{\emptyset,\mathcal{Q}\cap\mathcal{P}^{\mathcal{I}}}(s'')$, and $a.\overline{s} \xrightarrow{b} s''$. It follows that $b = a$, $s'' = \overline{s}$, and, therefore $s = \partial_{\emptyset,\mathcal{Q}}(a.\overline{s}) \xrightarrow{a} \partial_{\emptyset,\mathcal{Q}\cup\mathcal{P}^{\mathcal{I}}}(\overline{s})$. Now, by (rl_1) it holds that $t = a.\partial_{\mathcal{Q}\cap\mathcal{P}^{\mathcal{I}}}(\overline{s}) \xrightarrow{a} \partial_{\mathcal{Q}\cap\mathcal{P}^{\mathcal{I}}}(\overline{s})$. Similarly for $t \xrightarrow{b} t'$.
– If $P(s = \partial_{\emptyset,\mathcal{Q}}(a.\overline{s}))$, then $\overset{(rl_9)}{\Longrightarrow} P \notin \mathcal{Q}$ and $P(a.\overline{s}) \overset{(rl_7)}{\Longrightarrow} P \in \mathcal{P}^{\mathcal{I}}$ and $P\overline{s} \overset{(rl_9)}{\Longrightarrow} P \in \mathcal{P}^{\mathcal{I}}$ and $P(\partial_{\mathcal{Q}\cap\mathcal{P}^{\mathcal{I}}}(\overline{s})) \overset{(rl_7)}{\Longrightarrow} P(a.\partial_{\mathcal{Q}\cap\mathcal{P}^{\mathcal{I}}}(\overline{s}) = t)$. Similarly for Pt.

Therefore, by Lemma 7, it follows that $s \sim t$.

Take (A_{12}). Let $s = \partial_{\mathcal{B},\mathcal{Q}}(s_1 + s_2)$. Prove that $s \sim t = \partial_{\mathcal{B},\mathcal{Q}}(s_1) + \partial_{\mathcal{B},\mathcal{Q}}(s_2)$.

– If $s \xrightarrow{a} s'$ for some fixed $a \in Act$ and $s' \in S$, then by (rl_8), it follows that $a \notin \mathcal{B}$, $s' = \partial_{\emptyset,\mathcal{Q}}(s'')$, and $s_1 + s_2 \xrightarrow{a} s''$. Then $s_1 \xrightarrow{a} s''$ or $s_2 \xrightarrow{a} s''$ (by $(rl_{2,3})$), so $\partial_{\mathcal{B},\mathcal{Q}}(s_1) \xrightarrow{a} \partial_{\emptyset,\mathcal{Q}}(s'')$ or $\partial_{\mathcal{B},\mathcal{Q}}(s_2) \xrightarrow{a} \partial_{\emptyset,\mathcal{Q}}(s'')$ (by (rl_8)). Finally $\overset{(rl_{2,3})}{\Longrightarrow} \partial_{\mathcal{B},\mathcal{Q}}(s_1) + \partial_{\mathcal{B},\mathcal{Q}}(s_2) \xrightarrow{a} s'$. Similarly for $t \xrightarrow{a} t'$.
– Also, if Ps for some fixed $P \in \mathcal{P}$, then by (rl_9) it follows that $P \notin \mathcal{Q}$ and $P(s_1 + s_2)$. Then by $(rl_{2,3})$ and (rl_9) it follows that $P(\partial_{\mathcal{B},\mathcal{Q}}(s_1))$ or $P(\partial_{\mathcal{B},\mathcal{Q}}(s_2))$. Finally $\overset{(rl_{2,3})}{\Longrightarrow} P(\partial_{\mathcal{B},\mathcal{Q}}(s_1 + s_2))$. Similarly for Pt.

Therefore, by Lemma 7, it follows that $s \sim t$.

We have covered all the possibilities, so E^{∂}_{FTP} is sound for bisimilarity on $T(\Sigma^{\partial}_{FTP})$.

Proving $\forall t \in T(\Sigma^{\partial}_{FTP}), \exists t' \in T(\Sigma_{FTP})$ *s.t.* $E^{\partial}_{FTP} \vdash t = t'$

Note that in order to prove the statement above, it suffices to show that E^{∂}_{FTP} can be used to eliminate all the occurrences of ∂ from $\partial_{\mathcal{B},\mathcal{Q}}(t)$, for all closed terms $t \in T(\Sigma_{FTP})$, where $\mathcal{B}, \mathcal{Q}$ are two sets of restricted actions and, respectively, predicates. We proceed by structural induction on the structure of t.

Base cases

- $t = \delta \overset{(A_6)}{\Longrightarrow} E^{\partial}_{FTP} \vdash \partial_{\mathcal{B},\mathcal{Q}}(\delta) = \delta$;
- $t = c_P \overset{(A_{7,8})}{\Longrightarrow} E^{\partial}_{FTP} \vdash \partial_{\mathcal{B},\mathcal{Q}}(c_P) = \begin{cases} \delta & \text{if } P \in \mathcal{Q} \\ c_P & \text{if } P \notin \mathcal{Q} \end{cases}$.

Inductive step cases

- $t = t_1 + t_2 \overset{(A_{12})}{\Longrightarrow} \partial_{\mathcal{B},\mathcal{Q}}(t_1 + t_2) = \partial_{\mathcal{B},\mathcal{Q}}(t_1) + \partial_{\mathcal{B},\mathcal{Q}}(t_2)$. Both t_1 and t_2 are subterms of t, so, by the inductive hypothesis, $\exists t_1', t_2' \in T(\Sigma_{FTP})$ such that $E^{\partial}_{FTP} \vdash \begin{cases} \partial_{\mathcal{B},\mathcal{Q}}(t_1) = t_1' \\ \partial_{\mathcal{B},\mathcal{Q}}(t_2) = t_2' \end{cases} \overset{\text{congr.} \vdash}{\Longrightarrow} E^{\partial}_{FTP} \vdash \partial_{\mathcal{B},\mathcal{Q}}(t_1) + \partial_{\mathcal{B},\mathcal{Q}}(t_2) = t_1' + t_2' \in T(\Sigma_{FTP})$;
- $t = a.t'$
  - $a \in \mathcal{B} \overset{(A_9)}{\Longrightarrow} \partial_{\mathcal{B},\mathcal{Q}}(a.t') = \sum_{P \notin \mathcal{Q}, P(a.x)} c_P \in T(\Sigma_{FTP})$;
  - $a \notin \mathcal{B} \overset{(A_{10})}{\Longrightarrow} \partial_{\mathcal{B},\mathcal{Q}}(a.t') = \partial_{\emptyset,\mathcal{Q}}(a.t') \overset{(A_{11})}{=} a.\partial_{\emptyset,\mathcal{Q}\cap\mathcal{P}^{\mathcal{I}}}(t')$. As t' is a subterm of t, by the inductive hypothesis it follows that $\exists t'' \in T(\Sigma_{FTP})$ s.t. $E^{\partial}_{FTP} \vdash a.\partial_{\mathcal{Q}\cap\mathcal{P}^{\mathcal{I}}}(t') = a.t'' \in T(\Sigma_{FTP})$.

We have covered all the possibilities, so E^{∂}_{FTP} eliminates all the occurrences of ∂ from terms in $T(\Sigma^{\partial}_{FTP})$. □

G From general *preg* to smooth and distinctive

Given an arbitrary *preg* l-ary operation f, producing a family of smooth and distinctive operations capturing the behaviour of f consists of the following steps.

Transforming general *preg* to smooth operations. According to Definition 6, an operation f is not smooth if there exists (at least) one rule ρ for f that matches (at least) one of the following conditions:

- there are clashes between the sets I^+, I^-, J^+ and J^-,
- there is a positive position that has multiple transitions or that satisfies multiple predicates or
- ρ is a transition rule that contains in its target variables on positive positions.

By way of example, a rule satisfying all the items above is:

$$\rho = \frac{x \xrightarrow{a} y_1 \quad x \xrightarrow{b} y_2 \quad x \overset{d}{\not\to} \quad P_1 x \quad P_2 x \quad \neg P_3 x}{f(x) \xrightarrow{c} x + y_1}$$

Given a non-smooth operation f, identifying a smooth function f' that captures the behaviour of f implies "smoothening" all the rules of f. For the rule ρ given above, a convenient smooth rule ρ' is:

$$\rho' = \frac{x_1 \xrightarrow{a} y_1 \quad x_2 \xrightarrow{b} y_2 \quad x_3 \overset{d}{\nrightarrow} \quad P_1 x_4 \quad P_2 x_5 \quad \neg P_3 x_6}{f'(x_1, x_2, x_3, x_4, x_5, x_6, x_7) \xrightarrow{c} x_7 + y_1}$$

Note that the equation $f(x) = f'(x, x, x, x, x, x, x)$ holds and allows us to relate the behaviour of f with the one of its smoothened version.

Let G be an *preg* system and f a non-smooth l-ary function for G. The smoothening mechanism consists of the following steps (similar to those in [2]):

1. *Determine l' – the arity of f', the smooth version of f.* We start by counting how many fresh variables are needed for each position i in $\{1, \dots, l\}$.
   For each *preg* rule ρ for f (matching the format in Definition 1) and each i in $\{1, \dots, l\}$, we first define a "barb" value $B(\rho, i)$, as follows:
   $$B(\rho, i) = |I_i^+| + |J_i^+| + C_1 + C_2$$
   where
   $$C_1 = \begin{cases} 0, & \text{if } \mathcal{B}_i = \mathcal{Q}_i = \emptyset \\ 1, & \text{if } (\mathcal{B}_i = \emptyset \text{ and } \mathcal{Q}_i \neq \emptyset) \text{ or } (\mathcal{B}_i \neq \emptyset \text{ and } \mathcal{Q}_i = \emptyset) \\ 2, & \text{if } \mathcal{B}_i \neq \emptyset \text{ and } \mathcal{Q}_i \neq \emptyset \end{cases}$$
   and
   $$C_2 = \begin{cases} 0, & \text{if } x_i \text{ is not in the target of the conclusion} \\ 1, & \text{if } x_i \text{ is in the target of the conclusion} \end{cases}$$
   Note that if ρ is smooth, then $(\forall i \in \{1, \dots, l\}) \,.\, B(\rho, i) \leq 1$. Symmetrically, if ρ is non-smooth, then $(\exists i \in \{1, \dots, l\}) \,.\, B(\rho, i) > 1$.
   The number of variables needed for each $i \in \{1, \dots, l\})$ is
   $B(f, i) = max\{B(\rho, i) \mid \rho \text{ is a rule for } f\}$.
   We conclude that the arity of the function f' is $l' = \sum_{i=\overline{1,l}} B(f, i)$.
2. *Determine the smooth rules ρ' for f', based on the rules ρ for f.* Consider ρ a rule for f, $\vec{w} = w_{11}, \dots, w_{1B(f,1)}, \dots, w_{l1}, \dots, w_{lB(f,l)}$ a vector of l' different fresh variables (that do not occur in ρ), and consider a substitution $\tau[w_{ij} \mapsto x_i]$ mapping each w_{ij} to $x_i, i \in \{1, \dots, l\}, j \in \{1, \dots, n_i\}$. Let ρ' be a smooth *epreg* rule with the principal operation f', such that with the exception of their sources (resp. tests, for the case of predicate rules) $f(\vec{x})$ and $f'(\vec{w})$, $\rho'\tau[w_{ij} \mapsto x_i]$ and ρ are identical (note that this rule always exists). Then ρ' is the smooth version of f.
3. *Show that f and f' have the same behaviour.* This result is a consequence of the following lemma (similar to Lemma 4.12 in [2]):

   Lemma 8. *Suppose G is a* preg *system, and $S = f(\vec{z})$ and $S' = f'(\vec{v})$ are terms in $\mathbb{T}(\Sigma_G)$ with variables that do not occur in $\mathcal{R}_G$. Suppose that there exists a 1-1 correspondence between rules for f and rules for f' such that, whenever a rule ρ for f is related to a rule ρ' for f', we have that $\rho\langle\vec{z}/\vec{x}\rangle$ and $\rho'\langle\vec{v}/\vec{y}\rangle$ are identical with exception of:*
   – *their sources $f(\vec{x})$ and, respectively, $f'(\vec{y})$, if ρ and ρ' are transition rules*

– their tests $f(\vec{x})$ and, respectively, $f'(\vec{y})$, if ρ and ρ' are predicate rules
Then

$$f(\vec{z}) = f'(\vec{v})$$

is sound for bisimulation on $T(\Sigma_G)$.

The proof is a slightly modified version of the proof in [2]:

Proof. Consider $G' \sqsupseteq G$ and σ a closed $\Sigma_{G'}$–substitution. We have to prove that $s = f(\vec{z})\sigma \sim f'(\vec{v})\sigma = s'$, *i.e.*:
(a) $(\forall P \in \mathcal{P}) \,.\, Ps \Leftrightarrow Ps'$.
(b) $(\forall a \in \mathcal{A}) \,.\, s \xrightarrow{a} t \Leftrightarrow s' \xrightarrow{a} t$;
Case (a) "$\Rightarrow$". Assume Ps. We prove that Ps' holds.
As $\ltimes_{G'}, \rightarrow_{G'}$ are supported by G', $\mathcal{R}_{G'}$ contains a rule $\rho = \frac{H}{P(f(\vec{x}))}$ and there exists a $\Sigma_{G'}$–substitution τ s.t.:
(1) $\ltimes_{G'}, \rightarrow_{G'}, \tau \models H$
(2) $f(\vec{x})\tau = s$
We know that G' is a disjoint extension of G, therefore ρ is also a rule of G.
So, there is $\rho' = \frac{H'}{P(f'(\vec{y}))}$ a rule of $\mathcal{R}_G \subseteq \mathcal{R}_{G'}$ s.t. $H\langle\vec{z}/\vec{x}\rangle = H'\langle\vec{v}/\vec{y}\rangle$ (*).
As in [2], Lemma 4.12, we consider τ' a $\Sigma_{G'}$–substitution defined as follows:

$$\tau'(w) = \begin{cases} \sigma(w) & \text{if } w \text{ occurs in } \vec{z} \text{ or } \vec{v} \\ \tau(w) & \text{otherwise} \end{cases}$$

and claim that for all variables w that do not occur in $\vec{z}$ or $\vec{v}$, $(\tau' \circ \langle\vec{z}/\vec{x}\rangle)(w) = \tau(w)$. At this point we infer:
(1) $\Rightarrow$ $\ltimes_{G'}, \rightarrow_{G'}, \tau \models H$ $\Rightarrow$(variables of H do not occur in $\vec{z}$ or $\vec{v}$)
$\ltimes_{G'}, \rightarrow_{G'}, \tau' \circ \langle\vec{z}/\vec{x}\rangle \models H$ $\Rightarrow$(general property of $\models$)
$\ltimes_{G'}, \rightarrow_{G'}, \tau' \models H\langle\vec{z}/\vec{x}\rangle$ $\Rightarrow$(by (*))
$\ltimes_{G'}, \rightarrow_{G'}, \tau' \models H'\langle\vec{v}/\vec{y}\rangle$ $\Rightarrow$(general property of $\models$)
$\ltimes_{G'}, \rightarrow_{G'}, \tau' \circ \langle\vec{v}/\vec{y}\rangle \models H'$ $\Rightarrow$(soundness of $\ltimes_{G'}$ for ρ')
$\ltimes_{G'}, \rightarrow_{G'}, \tau' \circ \langle\vec{v}/\vec{y}\rangle \models P(f'(\vec{y}))$ $\Rightarrow$
$\ltimes_{G'}, \rightarrow_{G'}, \tau' \models P(f'(\vec{v}))$ $\Rightarrow$(def. $\models$)
$(P, f'(\vec{v})\tau') \in \ltimes_{G'}$ $\Rightarrow$($\tau' = \sigma$ on $\vec{v}$)
$(P, f'(\vec{v})\sigma) \in \ltimes_{G'}$, so Ps' also holds.

For the case (b) "$\Rightarrow$", the reasoning is the same as in [2], Lemma 4.12.
Cases (a) "$\Leftarrow$" and (b) "$\Leftarrow$" are symmetric. □

The following lemma is a straightforward result of the smoothening procedure presented in steps (1.)–(3.):

Lemma 9. *Consider G a* preg *system and f a non-smooth l-ary operation for G. Then there exists $G' \sqsupseteq G$ with f' a smooth l'-ary operation for some l', and there exist two vectors of variables $\vec{z}$ and $\vec{v}$ such that*

$$f(\vec{z}) = f'(\vec{v})$$

is sound for bisimulation on $T(\Sigma_{G''})$, for all the preg *systems G'' disjointly extending G'.*

Transforming smooth to smooth and distinctive operations. Given a smooth, but not distinctive, operation f', we next identify a family of smooth and distinctive operations $f'_1, \ldots, f'_n$ that "capture the behaviour of f'".

The procedure is identical to the one described in [2], Section 4.1.5. We consider $\mathcal{R}^1_G, \ldots, \mathcal{R}^n_G$, a partitioning of the set of rules for f' such that for all $i \in \{1, \ldots, n\}$, f' is distinctive in the *preg* system $(\Sigma_G, \mathcal{R}_G \setminus \bigcup_{\substack{j=\overline{1,n} \\ j \neq i}} \mathcal{R}^j_G)$. Note that this partitioning always exists; in the worst case each partition would be a singleton set.

Consider $\Sigma'_G = \Sigma_G \cup \{f'_1, \ldots, f'_n\}$, where $f'_1, \ldots, f'_n$ are fresh l'-ary operation symbols. Consider $\mathcal{R}'_G = \mathcal{R}_G \cup \{\widetilde{\mathcal{R}}^i_G \mid i \in \{1, \ldots, n\}\}$, where $\widetilde{\mathcal{R}}^i_G$ is obtained from $\mathcal{R}^i_G$ by replacing f' in the source with f'_i. Let $G' = (\Sigma'_G, \mathcal{R}'_G)$. Note that G' is a disjoint extension of G. It is trivial to check that

$$f'(\vec{x}) = f'_1(\vec{x}) + \ldots + f'_n(\vec{x})$$

is sound for bisimilarity on $T(\Sigma'_G)$.

Lemma 10. *Consider G a* preg *system and f a smooth l-ary operation for G. Then there exist $G' \sqsupseteq G$ and $f_1, \ldots, f_n$ smooth and distinctive l-ary operations for G' such that*

$$f(\vec{x}) = f_1(\vec{x}) + \ldots + f_n(\vec{x})$$

is sound for bisimulation on $T(\Sigma_{G''})$, for all the preg *systems G'' disjointly extending G'.*

H A possible approach to handle implicit predicates

So far the current framework handles implicit predicates up to trees (*i.e.*, terms over the grammar (1)). For the case of arbitrary *preg* operations f we encountered some problems when proving the soundness of the action law $f(\vec{X}) = c.C[\vec{X}, \vec{y}]$ used in the head-normalization process.

Consider the closed instantiation $\vec{X} = \vec{s}$, $\vec{y} = \vec{t}$ and assume that $P(f(\vec{s}))$ holds for some predicate P in $\mathcal{P}^{\mathcal{I}}$. Then $P(c.C[\vec{s}, \vec{t}])$ should also hold, so we should have $P(C[\vec{s}, \vec{t}])$. One possible way of ensuring this property syntactically would be to stipulate some consistency requirements.

Consider, for instance, a scenario in which a transition rule for f has the conclusion $f(\vec{X}) \xrightarrow{c} C[\vec{X}, \vec{y}]$ and there is a predicate rule $\frac{H}{P(f[\vec{X}])}$. If there is a derivable predicate rule (ruloid) $\frac{H'}{P(C[\vec{X},\vec{y}])}$ with $H' \subseteq H$ then this would ensure that if the left-hand side of the action law satisfies P then so does then right-hand side. A symmetrical consistency requirement can be formulated to ensure that $P(f(\vec{s}))$ holds whenever $P(c.C[\vec{s}, \vec{t}])$ holds.

Note that for certain restricted *preg* systems these consistency requirements are guaranteed by the format of the rules. This is the case of the *preg* transition rules with "CCS-like" conclusions, *i.e.*, $C[\vec{X}, \vec{y}]$ is a variable or it is of the form $g(z_1, ..., z_m)$.

I Proof of Theorem 3

Proving [soundness]

Proof. Let $f \in \Sigma_G$ be a smooth and distinctive operation with the arguments $\vec{t} = t_1, \ldots, t_l$ as closed terms over Σ_G.

Distributivity law.

Let $i \in L$ be a positive position for f (every rule for f tests i positively) with $t_i = t'_i + t''_i$. We want to argue that the distributivity axiom for f is sound for position i.

Assume $\exists \rho \in \mathcal{R}^{\mathcal{A}}$ (of the form (1)) such that $f(t_1, \ldots, t'_i + t''_i, \ldots, t_l) \xrightarrow{c} t$ for some t. Then there are an action rule ρ and a closed substitution σ such that (a) σ satisfies the premises of ρ and (b) $f(t_1, \ldots, t'_i + t''_i, \ldots, t_l) \xrightarrow{c} t$ is obtained by instantiating the conclusion of ρ with σ. Since f is smooth and distinctive, i is a positive position for f, and ρ is a rule for f, we infer that there is a positive premise for i in ρ, i.e. $i \in I^+_\rho \cup J^+_\rho$. In either case, we may assume, without loss of generality, that the positive premise is satisfied by t'_i. Take $\sigma' = \sigma[x_i \mapsto t'_i]$. Then σ' satisfies the premises of ρ and proves that $f(t_1, \ldots, t'_i, \ldots, t_l) \xrightarrow{c} t$. This is where we actually make use of the restriction for x_i to not occur in the target of the conclusion of ρ (Definition 6). Therefore, the right hand side of the instance of the distributivity law also has the transition to t that performs c.

We follow a similar reasoning for $f(t_1, \ldots, t'_i, \ldots, t_l) + f(t_1, \ldots, t''_i, \ldots, t_l) \xrightarrow{c} t$.

The case in which $\exists \rho \in \mathcal{R}^{\mathcal{P}}$ (of the format (2)) such that $P(f(t_1, \ldots, t'_i + t''_i, \ldots, t_l))$ is also handled similarly (it is actually easier).

We conclude that

$$f(t_1, \ldots, t'_i + t''_i, \ldots, t_l) \sim f(t_1, \ldots, t'_i, \ldots, t_l) + f(t_1, \ldots, t''_i, \ldots, t_l).$$

Action law. Let ρ be a transition rule (of the format (1)) for f s.t. $t_1, \ldots, t_l$ match the restrictions the over the arguments of f in Definition 8.2, and t_c be the corresponding closed instantiation of $C[\vec{X}, \vec{y}]$.

Obviously $c.t_c$ can only perform action c and reach t_c, and $c.t_c$ does not satisfy any predicate. By the hypothesis we know that ρ is a rule of f, so f can also perform c and reach the same t_c.

Next we want to prove that ρ is the only rule that applies for f (*i.e.*, $f(\vec{t})$ cannot perform any other action and does not satisfy predicates). Assume there exits $\rho' \neq \rho$ for f, that can be applied. As f is smooth and distinctive it follows that one of the following holds:

1. $\exists i \in I^+_\rho \cap I^+_{\rho'} \neq \emptyset$ s.t. $a_i \neq a'_i$ – case in which ρ' cannot be applied, since $t_i = a_i.t'_i$ can only perform action a_i;
2. $\exists i \in J^+_\rho \cap J^+_{\rho'} \neq \emptyset$ s.t. $P_i \neq P'_i$ – case in which ρ' cannot be applied, since $t_i = c_{P_i}$ only satisfies P_i;
3. $\exists i \in I^+_\rho \cap J^+_{\rho'} \neq \emptyset$ – but this cannot be the case since $t_i = a_i.t'_i$ does not satisfy any predicate;

4. $\exists i \in J^+_\rho \cap I^+_{\rho'} \neq \emptyset$ – but this cannot be the case since $t_i = c_{P_i}$ does not perform any action.

We thus reached a contradiction.

We conclude that
$$f(\vec{t}) \sim c.t_c \ .$$

Predicate law. Let ρ be a predicate rule (of the format (2)) for f s.t. $t_1, \dots, t_l$ match the restrictions over the arguments of f in Definition 8.2.

Obviously that c_P does not perform any action, and only satisfies P. By following a similar reasoning to the one for the case of *action laws*, we show that there is no rule $\rho' \neq \rho$ that applies for f. So, $f(\vec{t})$ does not perform any action, and does not satisfy any predicate besides P.

We conclude that
$$f(\vec{X}) \sim c_P.$$

Deadlock law. The soundness follows immediately from the restrictions imposed by Definition 8.3.

$\square$

Proving head normalization

Proof. The result is an immediate consequence of the following claim:

Claim Let $f \in \Sigma$ be a smooth and distinctive operation with the arguments $\vec{t} = t_1, \dots, t_l$ as closed terms over Σ_G in head normal form. Then there exists a closed term t over Σ_G in head normal form such that $E_G \vdash f(\vec{t}) = t$.

The proof follows by induction on the combined size of $t_1, \dots, t_l$. We perform a case analysis:

1. $(\exists i \in L$ positive for $f)$. $t_i = t'_i + t''_i$. We apply the distributivity law and the inductive hypothesis.
2. $(\exists i \in L$ positive for $f)$. $t_i = \delta$. We apply a deadlock law.
3. $(\forall i \in L$ positive for $f)$. t_i is either of the form $a_i.t'_i$ or c_{P_i}.
   (a) $(\forall \rho \in \mathcal{R}$ for $f)$. $((\exists j \in I^+_\rho)$. $t_j = a'_j.t'_j (a'_j \neq a_j)$ or $t_j = c_{P'_j})$ or $((\exists j \in J^+_\rho)$. $t_j = a'_j.t'_j$ or $t_j = c_{P'_j}(P'_j \neq P_j)) \Longrightarrow$ we can apply a deadlock law.
   (b) $((\exists \rho \in \mathcal{R}$ for $f)$. $((\forall j \in I^+_\rho)$. $t_j = a_j.t'_j)$ and $((\forall j \in J^+_\rho)$. $t_j = c_{P_j})$.
   Remark that, by the distinctiveness of f, ρ is the only rule that could be fired ($\bullet$). In order to prove this, assume, for instance, $\exists \rho' \neq \rho$ that could applied. If $I^+_\rho = J^+_\rho = \emptyset$ then this comes in contradiction with the second item of Definition 7. On the other hand, if $I^+_\rho \neq \emptyset$ then one of the following holds:
      – $(\exists k \in I^+_\rho \cap I^+_{\rho'} \neq \emptyset)$. $a_k \neq a'_k$ – case in which ρ' cannot be applied, since t_k can only fire a_k;

– $\exists k \in I^+_\rho \cap J^+_{\rho'} \neq \emptyset$ – case in which we reach a contradiction, since by hypothesis we know that $(\forall j \in I^+_\rho) \,.\, t_j = a_j.t'_j$ which does not satisfy any predicate.

The reasoning is similar for $J^+_\rho \neq \emptyset$. So ρ is the only rule that can be applied for f.

We further identify the following situations:

i. $(\exists k \in I^-_\rho) \,.\, t_k = b.t'_k + t''_k \, (b \in \mathcal{B})$ or $(\exists k \in J^-_\rho) \,.\, t_k = t'_k + c_Q \, (Q \in \mathcal{Q}_k)$ $\overset{(\bullet)}{\Longrightarrow}$ we can apply a deadlock law.

ii. $(\forall k \in I^-_\rho \cup J^-_\rho) \,.\, t_k = \sum_{i \in I} a^i_k.t^i_k + \sum_{j \in J} c_{P^j_k}$ s.t. $\{a^i_k \mid i \in I\} \cap \mathcal{B}_k = \emptyset$ and $\{P^j_k \mid j \in J\} \cap \mathcal{Q}_k = \emptyset$.
Recall the operator ∂ is used to specify restrictions on the actions performed and/or on the predicates satisfied by a given term. Therefore, for a term t_k satisfying the conditions in the hypothesis, it is intuitive to see that $t_k = \partial_{\mathcal{B},\mathcal{Q}}(t_k)$. In what follows, we provide an auxiliary result formalizing this intuition:

Lemma 11. *Consider $G \sqsupseteq FTP^{\partial}$ a* preg *system and a term $t = \sum_{i \in I} a_i.t_i + \sum_{j \in J} c_{P_j} \in T(\Sigma_G)$ in h.n.f. Let $\mathcal{B}$ and $\mathcal{Q}$ be two sets of restricted actions and predicates, respectively. If $\{a_i \mid i \in I\} \cap \mathcal{B} = \emptyset$ and $\{P_j \mid j \in J\} \cap \mathcal{Q} = \emptyset$ then $E^{\partial}_{FTP} \vdash t = \partial_{\mathcal{B},\mathcal{Q}}(t)$.*

Proof. The proof follows immediately, by induction on the structure of t and by applying the axioms (A_6) – (A_{12}). □

In these conditions, according to Lemma 11, it holds that $(\forall k \in I^-_\rho \cup J^-_\rho) \,.\, t_k = \partial_{\mathcal{B}_k,\mathcal{Q}_k}(t_k)$. Therefore, all the requirements for applying a trigger law are met: if $\rho \in \mathcal{R}^{\mathcal{A}}$ then apply the action law for ρ, and if $\rho \in \mathcal{R}^{\mathcal{P}}$ then apply the predicate law for ρ.

This reasoning suffices to guarantee that E_G is head normalizing. □

J Proof of Lemma 4

Proof. Let $t, t' \in T(\Sigma_{FTP_/})$ s.t. $(\forall n \in \mathbb{N}) \,.\, t/c^n = t'/c^n$. We have $(\forall n \in \mathbb{N}) \,.\, t/c^n \sim t'/c^n$ and we want to prove that $t \sim t'$. In other words, we have to build a bisimulation relation $R \subseteq T(\Sigma_{FTP_/}) \times T(\Sigma_{FTP_/})$ s.t. $(t, t') \in R$.

First we make the following observations:

1. $\forall t \in T(\Sigma_{FTP_/})$ it holds that t is finitely branching (by Lemma 1);
2. if $t/c^n \sim t'/c^n$ then $(\forall P \in \mathcal{P}) \,.\, (Pt \Leftrightarrow Pt')$, where $t, t' \in T(\Sigma_{FTP_/})$ and $n \in \mathbb{N}$ (the reasoning is as follows: if $t/c^n \sim t'/c^n$ then by the definition of bisimilarity it holds that $P(t/c^n) \Leftrightarrow P(t'/c^n)$, so by (rl_{11}) it follows that $Pt \Leftrightarrow Pt'$).

Next we show that the relation $R \subseteq T(\Sigma_{FTP_/}) \times T(\Sigma_{FTP_/})$ defined as

$$(t, t') \in R \text{ iff } (\forall n \in \mathbb{N}) \,.\, t/c^n \sim t'/c^n$$

is a bisimulation relation.
Assume $(t, t') \in R$ and $t \xrightarrow{a} t_1$ and for $n > 0$ define $S_n = \{t^* \mid t' \xrightarrow{a} t^* \text{ and } t_1/c^n \sim t^*/c^n\}$. Note that:

- $S_1 \supseteq S_2 \supseteq \ldots$, as $t_1/c^{n+1} \sim t^*/c^{n+1} \Rightarrow t_1/c^n \sim t^*/c^n$ (the latter implication is straightforward by the definition of bisimilarity and by (rl_{10}), (rl_{11}));
- $(\forall n > 0) \,.\, S_n \neq \emptyset$, since by the definition of R we have that $t_1/c^{n+1} \sim t^*/c^{n+1}$, and $t \xrightarrow{a} t_1$ according to the hypothesis;
- each S_n is finite, as t' is finitely branching (1.).

At this point we conclude that the sequence $S_1, S_2, \ldots, S_n, \ldots$ remains constant from some n onwards. Therefore $\cap_{n>0} S_n \neq \emptyset$. Let t' be an element of this intersection. We have that: $t' \xrightarrow{a} t'_1$, $(t_1, t'_1) \in R$ and $(\forall P \in \mathcal{P}) \,.\, Pt \Rightarrow_{(2.)} Pt'$ ($\clubsuit$). We follow a similar reasoning for the symmetric case $t' \xrightarrow{a} t'_1$ and show that $\exists t_1$ s.t. $t \xrightarrow{a} t_1$, for $(t_1, t'_1) \in R$ and $(\forall P \in \mathcal{P}) \,.\, Pt' \Rightarrow_{(2.)} Pt$ ($\spadesuit$). By ($\clubsuit$) and ($\spadesuit$) it follows that R is a bisimulation relation, so $(t, t') \in R \Rightarrow t \sim t'$. $\square$