Theoretical Computer Science 411 (2010) 1338–1371

Contents lists available at ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

Invariants for Parameterised Boolean Equation Systems

Simona Orzan, Tim A.C. Willemse *

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

ARTICLE INFO

Article history:
Received 17 October 2008
Received in revised form 4 August 2009
Accepted 4 November 2009
Communicated by R. van Glabbeek

Keywords:
Parameterised Boolean Equation Systems
Invariants
Fixed point theory
Model checking
Process theory

ABSTRACT

The concept of invariance for Parameterised Boolean Equation Systems (PBESs) is studied in greater detail. We identify an issue with the associated theory and fix this problem by proposing a stronger notion of invariance called global invariance. A precise correspondence is proven between the solution of a PBES and the solution of its invariant-strengthened version; this enables one to exploit global invariants when solving PBESs. Furthermore, we show that global invariants are robust w.r.t. all common PBES transformations and that the existing encodings of verification problems into PBESs preserve the invariants of the processes involved. These traits provide additional support for our notion of global invariants, and, moreover, provide an easy manner for transferring (e.g. automatically discovered) process invariants to PBESs. We provide several examples that illustrate the use of global invariants for a variety of verification problems.

1. Introduction

Parameterised Boolean Equation Systems (PBESs), introduced in [21,20], and studied in more detail in [15] are sequences of fixed point equations of the form $\sigma X(d{:}D) = \phi$, where $\sigma \in \{\mu, \nu\}$ is a fixed point sign, X is a predicate variable, ϕ a predicate formula in which predicate variables may occur, and d of sort D is a data variable that may occur in ϕ. Each equation defines a solution for its predicate variable; these solutions are functions from some domain D to the Booleans. In general, the solution of a predicate variable X recursively depends on the solution of predicate variables that are defined by equations in the PBES (i.e. including the equation for X itself).

Over the course of the past decade, PBESs have been used for studying and solving a variety of verification problems for complex reactive systems. Problems as diverse as model checking problems for symbolic transition systems [11,14] and real-time systems [29]; equivalence checking problems for a variety of process equivalences [4]; and static analysis of code [9] have been encoded in the PBES framework. The solution to these encoded problems can be found by computing the truth of a predicate formula which has to be interpreted in the context of the solution to the PBES. Several verification tools rely on PBESs or fragments thereof, e.g. the μCRL [14] and the mCRL2 [5] model checkers and the CADP toolsuite [10].

Solving a PBES is in general an undecidable problem, much like the problems that can be encoded in them. Nevertheless, there are pragmatic approaches to solving PBESs, such as *symbolic approximation* [15] and *instantiation* [5]; the latter tries to compute a *Boolean Equation System* (BES) [19], which is part of a fragment of PBESs for which the problem of computing the solution is decidable. While these techniques have proved their merits in practice, the undecidability of solving PBESs in general implies that these techniques are not universally applicable.

A concept that has turned out to be very powerful, especially in combination with symbolic approximation is the notion of an *invariant* for PBESs. For instance, invariants have been used successfully in [4] when solving PBESs encoding the branching bisimulation problem for two systems: the invariants allowed the symbolic approximation process to terminate in a few

* Corresponding author. Tel.: +31 402472952.
E-mail addresses: T.A.C.Willemse@tue.nl, timw@bijdehand.org (T.A.C. Willemse).

0304-3975/$ – see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2009.11.001

steps, whereas there was no indication that it could have terminated without the invariant. As such, the notion of an invariant is a powerful tool which adds to the efficacy of techniques and tooling as described in [14].

An invariant for a PBES, as defined in [15] (hereafter referred to as a *local invariant*), is a relation on data variables of a PBES that provides an over-approximation of the dependencies of the solution of a particular predicate variable X on its own domain. Unfortunately, the theory of local invariants as outlined in [15], is too ungainly for *arbitrary* equation systems.

We show that using a local invariant in combination with standard PBES manipulations can wrongfully affect the solution to a PBES. This situation is remedied by introducing the concept of a *global invariant*, and we show how this notion relates to local invariants. Moreover, we demonstrate that global invariants are preserved by common solution-preserving PBES manipulation methods, viz. *unfolding*, *migration* and *substitution* [15]. An invariance theorem that allows one to calculate the solution for an equation system, using a global invariant to assist the calculation, is proved. As a side-result of our invariance theorem, we are able to provide a partial answer to a generalisation of an open problem coined in [15], which concerns the solution to a particular PBES pattern. Patterns are important as they allow for a simple *look-up* and *substitute* strategy to solving a PBES. Finally, we prove that traditional *process invariants* [2] are preserved under the PBES encoding of the first-order modal μ-calculus model checking problem [14] and the PBES encoding of all four process equivalences that are described in [4], viz. strong-, branching- and weak bisimulation and (branching) simulation equivalence. From a practical viewpoint, the preservation of process invariants under these encodings is important, as this avoids computing the solution for the PBES for states that cannot be reached (which is a major cause for non-termination of symbolic approximation).

To illustrate the efficacy of using invariants for verifications conducted within the PBES framework, we provide several examples, including a verification of a *Cache Coherence Protocol* from the literature [1,25]. The examples vary in complexity, and illustrate various types of verification problems. Many examples involve parametric systems, meaning that the verifications are conducted over *all instances* of these systems.

Note that an abstract of this paper appeared as [22]. Apart from basic lemmata that support the detailed proofs of all propositions and theorems, this paper contains additional results aiding in the understanding of the characteristics of (global) invariants for PBESs, new results stating that process invariants for three equivalences other than branching bisimilarity are preserved, and several additional examples demonstrating the use of invariants for PBESs in conducting process verifications.

Related work. The concept of an *invariant*, first defined by Floyd [7], has been indispensable in many complex verification tasks. Traditionally, invariants have been employed for proving correctness of non-elementary sequential algorithms [7,16]; more recently, invariants have also been put to use in the verification of distributed and concurrent systems. In the latter area, correctness has a different flavour, but invariants fulfill the role of characterising reachability of states, facilitating or even enabling property verification. Verification using PBESs, and model checking in particular, has the advantage that it encodes only the process behaviour that is important for the property at hand; as such, a PBES can have invariants that are not invariants for the original process (see Section 6 for an example illustrating this point).

Historically, the main use of invariants is in proofs of safety properties like data consistency or mutual exclusion [2,24]; liveness properties, on the other hand, are better supported by variant notions like the ranking functions [3,6]. These capture the monotonic dynamics of a property rather than its stability through process execution.

Invariants provide the foundation of many mature verification methodologies aiming to tackle complex cases, such as networks of parameterised systems [24,25,6], various types of equivalence checks between reactive system [2] and for infinite data domains in general, such as hybrid systems [26]. These research efforts are aimed at stretching the limits of verification for specific classes of systems and properties. In contrast, PBESs have the advantage that the techniques developed for them, including the invariance theorems of this paper, are universally applicable to all problems that can be encoded in them.

Several works, like [24,6,26] focus on the automated and even automatic discovery of invariants for specialised classes of specifications and properties. We suspect that many of these techniques can be ported to work for specific PBESs as well (although the actual porting may be rather involved). This is supported by our result that demonstrates that process invariants are preserved under the existing encodings of verification problems, meaning that any "discovered" process invariant immediately gives rise to a global invariant in the PBES that encodes some verification problem for the process at hand.

Structure. In Section 2, we introduce PBESs and some basic notation and results. We recall the definition of local invariants, and introduce global invariants in Section 3. In Section 4, we provide the main invariance theorem for global invariants, resolving the issue with the local invariance theorem. Robustness of the notion of a global invariant with respect to PBES is shown in Section 5. The relation between process invariants and global invariants is addressed in Section 6. Examples and applications of invariants for PBESs are provided in Section 7. Finally, in Section 8, we present our conclusions and provide pointers for future work.

2. Background

Parameterised Boolean Equation Systems are sequences of fixed point equations over predicate formulae. The latter are similar to first-order formulae in positive form. *Predicate variables* occurring in predicate formulae are used to represent

arbitrary formulae. In Section 2.1, we formalise the notion of predicate formulae; subsequently, in Section 2.2, we provide several results that allow us to reason about syntactic substitution in predicate formulae. Finally, we provide the syntax and semantics of Parameterised Boolean Equation Systems in Section 2.3, along with several known techniques for manipulating such systems.

2.1. Predicate formulae

Throughout this article, we assume that data sorts represent nonempty data types. As a convention, we write data sorts using letters D, E and F. In line with standard abstract data type theoretical approaches, we furthermore assume that a data sort specification consists of a sort declaration, constructor elements, operations and equations that state how the operations and constructors (and possibly other data sorts) are related.

We furthermore assume the existence of a sort $B = \{\top, \bot\}$, representing the Booleans, and the sort $N = \{0, 1, \ldots\}$ representing the natural numbers. For these sorts, we assume the usual operators are available.

Definition 1. A *predicate formula* is a formula ϕ in positive form, defined by the following grammar:

$$\phi ::= b \mid \phi_1 \wedge \phi_2 \mid \phi_1 \vee \phi_2 \mid \forall d{:}D.\ \phi \mid \exists d{:}D.\ \phi \mid X(\vec{e})$$

where b is a data term of Boolean sort B, possibly containing data variables $d \in \mathcal{D}$. Furthermore, X (taken from some domain of variables $\mathcal{P}$) is a (sorted) predicate variable to which we associate a vector of data variables $\vec{d_X}$ of sort $\vec{D_X}$; $\vec{e}$ is a vector of data terms of the sort $\vec{D_X}$; The data variables occurring in a predicate formula are taken from a set $\mathcal{D}$ of data variables.

Remark that throughout this paper, we assume that data terms do not contain predicate variables, i.e., $X(X(e))$ is not a valid predicate formula. The set of all predicate formulae is denoted Pred. Predicate formulae ϕ that do not contain predicate variables are referred to as *simple predicates*. The set of predicate variables that occur in a formula ϕ is denoted by $\mathrm{occ}(\phi)$.

Remark 2. Note that negation does not occur in predicate formulae, except as an operator in data terms. We use $b \implies \phi$ as a shorthand for $\neg b \vee \phi$ for terms b of sort B.

Remark 3. As usual, we use predicate variables X to which we associate a single variable d_X of sort D_X instead of vectors $\vec{d_X}$ of sort $\vec{D_X}$ in our definitions and theorems. This does not incur a loss of generality of the theory, as more complex formulae can be obtained using suitable pairing and projection functions.

Predicate formulae may contain both data variables that are bound by a universal/existential quantifier, and data variables that are free. We assume that the set of bound variables and the set of free variables in a predicate formula are disjoint. For a closed data term e, i.e. a data term not containing free data variables, we assume an interpretation function $[\![\_]\!]$ that maps the term e to the semantic data element $[\![e]\!]$ it represents. For open terms, we use a *data environment* ε that maps each variable from $\mathcal{D}$ to a data value of the intended sort. The interpretation of an open term e is denoted by $[\![e]\!]\,\varepsilon$ and is obtained in the standard way. We write $\varepsilon[e/d]$ to stand for the environment ε for all variables different from d, and $\varepsilon[v/d](d) = v$. A similar notation applies to predicate environments.

Definition 4. Let θ be a *predicate environment* assigning a function of type $D_X \to B$ to every predicate variable X, and let ε be a *data environment* assigning a value from domain D to every variable d of sort D. The interpretation $[\![\_]\!]\,\theta\varepsilon$ of a predicate formula in the context of environment θ and ε is either true or false, determined by the following induction:

$$\begin{aligned}
[\![b]\!]\,\theta\varepsilon &= [\![b]\!]\,\varepsilon \\
[\![\phi_1 \wedge \phi_2]\!]\,\theta\varepsilon &= [\![\phi_1]\!]\,\theta\varepsilon \text{ and } [\![\phi_2]\!]\,\theta\varepsilon \\
[\![\phi_1 \vee \phi_2]\!]\,\theta\varepsilon &= [\![\phi_1]\!]\,\theta\varepsilon \text{ or } [\![\phi_2]\!]\,\theta\varepsilon \\
[\![\forall d{:}D.\ \phi]\!]\,\theta\varepsilon &= \text{for all } v \in D,\ [\![\phi]\!]\,\theta(\varepsilon[v/d]) \\
[\![\exists d{:}D.\ \phi]\!]\,\theta\varepsilon &= \text{for some } v \in D,\ [\![\phi]\!]\,\theta(\varepsilon[v/d]) \\
[\![X(e)]\!]\,\theta\varepsilon &= \text{true if } \theta(X)([\![e]\!]\,\varepsilon) \text{ and false otherwise}
\end{aligned}$$

Remark 5. We do not formally distinguish between the abstract sorts of data variables and predicate variables, and the semantic sets they represent.

We partially order predicate formulae by means of the semantic implication $\to$: a predicate formula ϕ *implies* a predicate formula ψ iff for any environment, the interpretation of ϕ implies the interpretation of ψ:

Definition 6. Let ϕ and ψ be predicate formulae. We write $\phi \to \psi$ iff for all predicate environments θ and all data environments ε, $[\![\phi]\!]\,\theta\varepsilon$ implies $[\![\psi]\!]\,\theta\varepsilon$.

The symmetric closure of $\to$ induces the *logical equivalence* on Pred, denoted $\leftrightarrow$. Basic properties such as commutativity, idempotence and associativity of $\wedge$ and $\vee$ are immediately satisfied.

2.2. Predicate variables and substitution

A basic operation on predicate formulae is substitution of a predicate formula for a predicate variable. To this end, we introduce *predicate functions*: predicate formulae casted to functions. As a shorthand, we write $\phi_{\langle d_X \rangle}$ to indicate that ϕ is lifted to a function $(\lambda d_X{:}D_X.\ \phi)$, i.e. $\phi_{\langle d_X \rangle}$ takes an expression e of sort D_X and yields the predicate ϕ in which all occurrences of d_X have been replaced by expression e. The semantics of such a predicate function is defined in the context of a predicate environment θ and a data environment ε:

$$[\![\phi_{\langle d_X \rangle}]\!]\,\theta\varepsilon = \lambda v \in D_X.\ [\![\phi]\!]\,\theta\varepsilon[v/d_X]$$

Lemma 7. *Let ϕ, ψ be arbitrary predicate formulae. We have $\phi \leftrightarrow \psi$ iff for all environments θ, ε, $[\![\phi_{\langle d_X \rangle}]\!]\,\theta\varepsilon = [\![\psi_{\langle d_X \rangle}]\!]\,\theta\varepsilon$.*

Proof. Follows by definition of $\leftrightarrow$. □

Syntactic substitution of a predicate function $\psi_{\langle d_X \rangle}$ for a predicate variable X in a predicate formula ϕ is formalised by the following set of rules:

$$\begin{array}{rcl}
b[\psi_{\langle d_X \rangle}/X] & = & b \\
Y(e)[\psi_{\langle d_X \rangle}/X] & = & \begin{cases} \psi[e/d_X] \text{ if } Y = X \\ Y(e) \quad \text{otherwise} \end{cases} \\
(\phi_1 \wedge \phi_2)[\psi_{\langle d_X \rangle}/X] & = & \phi_1[\psi_{\langle d_X \rangle}/X] \wedge \phi_2[\psi_{\langle d_X \rangle}/X] \\
(\phi_1 \vee \phi_2)[\psi_{\langle d_X \rangle}/X] & = & \phi_1[\psi_{\langle d_X \rangle}/X] \vee \phi_2[\psi_{\langle d_X \rangle}/X] \\
(\forall d{:}D.\ \phi)[\psi_{\langle d_X \rangle}/X] & = & \forall d{:}D.\ \phi[\psi_{\langle d_X \rangle}/X] \\
(\exists d{:}D.\ \phi)[\psi_{\langle d_X \rangle}/X] & = & \exists d{:}D.\ \phi[\psi_{\langle d_X \rangle}/X]
\end{array}$$

Example 8. Consider the predicate formulae $\phi := X(f(d)) \wedge Y(g(d))$ and $\psi := Y(h(d_Y))$. The syntactic substitution of predicate function $\psi_{\langle d_Y \rangle}$ for Y in ϕ yields:

$$\begin{array}{l}
\big(X(f(d)) \wedge Y(g(d))\big)[\psi_{\langle d_Y \rangle}/Y] \\
= X(f(d)) \wedge Y(g(d))[\psi_{\langle d_Y \rangle}/Y] \\
= X(f(d)) \wedge Y(h(g(d)))
\end{array}$$

The predicate environment, being a semantic entity, and the syntactic substitution, being an abstract operation on predicate formulae, are closely related. The exact correspondence is given by the following property.

Property 9. *Let ϕ, ψ be arbitrary predicate formulae and let X of sort D_X be a predicate variable. For all environments θ, ε, the following correspondence holds:*

$$[\![\phi[\psi_{\langle d_X \rangle}/X]]\!]\,\theta\varepsilon = [\![\phi]\!]\,\theta[\,[\![\psi_{\langle d_X \rangle}]\!]\,\theta\varepsilon\,/X]\varepsilon$$

Proof. Follows by an induction on the structure of ϕ. □

We have the following lemmata dealing with syntactic substitutions and logical equivalence. Apart from the additional insight into the subtle interactions between logical equivalence and substitutions one gains through these lemmata, they provide the necessary foundation for most of the proofs and theorems in the remaining sections.

Lemma 10. *Let ψ, ρ, χ be arbitrary predicate formulae. If $\psi \leftrightarrow \rho$ holds, then $\chi[\psi_{\langle d_X \rangle}/X] \leftrightarrow \chi[\rho_{\langle d_X \rangle}/X]$ holds.*

Proof. Let θ, ε be arbitrary environments. We show that the following implication holds:

$$[\![\psi]\!]\,\theta\varepsilon = [\![\rho]\!]\,\theta\varepsilon \quad \text{implies } [\![\chi[\psi_{\langle d_X \rangle}/X]]\!]\,\theta\varepsilon = [\![\chi[\rho_{\langle d_X \rangle}/X]]\!]\,\theta\varepsilon$$

From the assumption $\psi \leftrightarrow \rho$, it follows that $[\![\psi]\!]\,\theta\varepsilon = [\![\rho]\!]\,\theta\varepsilon$ holds. We continue our reasoning as follows:

$$\begin{array}{ll}
 & [\![\psi]\!]\,\theta\varepsilon = [\![\rho]\!]\,\theta\varepsilon \\
\Rightarrow & \{\text{Lemma 7}\} \\
 & \theta[\,[\![\psi_{\langle d_X \rangle}]\!]\,\theta\varepsilon/X] = \theta[\,[\![\rho_{\langle d_X \rangle}]\!]\,\theta\varepsilon/X] \\
\Rightarrow & \\
 & [\![\chi]\!]\,\theta[\,[\![\psi_{\langle d_X \rangle}]\!]\,\theta\varepsilon/X] = [\![\chi]\!]\,\theta[\,[\![\rho_{\langle d_X \rangle}]\!]\,\theta\varepsilon/X] \\
\Leftrightarrow & \{\text{Property 9}\} \\
 & [\![\chi[\psi_{\langle d_X \rangle}/X]]\!]\,\theta\varepsilon = [\![\chi[\rho_{\langle d_X \rangle}/X]]\!]\,\theta\varepsilon \quad \square
\end{array}$$

Lemma 11. *Let ψ, ρ, χ be arbitrary predicate formulae. If $\psi \leftrightarrow \rho$ holds then $\psi[\chi_{\langle d_X \rangle}/X] \leftrightarrow \rho[\chi_{\langle d_X \rangle}/X]$ holds.*

Proof. Let θ, ε be arbitrary environments. We demonstrate that:

$$[\![\psi[\chi_{\langle d_X\rangle}/X]]\!]\,\theta\varepsilon = [\![\rho[\chi_{\langle d_X\rangle}/X]]\!]\,\eta\varepsilon$$

This follows from the following reasoning:

$$\begin{aligned}
&[\![\psi[\chi_{\langle d_X\rangle}/X]]\!]\,\theta\varepsilon\\
=\ &\{\text{Property 9}\}\\
&[\![\psi]\!]\,\theta[\,[\![\chi_{\langle d_X\rangle}]\!]\,\theta\varepsilon/X]\varepsilon\\
=\ &\{\psi \leftrightarrow \rho, \text{so } \psi \text{ and } \rho \text{ are indistinguishable for all environments }\}\\
&[\![\rho]\!]\,\theta[\,[\![\chi_{\langle d_X\rangle}]\!]\,\theta\varepsilon/X]\varepsilon\\
=\ &\{\text{Property 9}\}\\
&[\![\rho[\chi_{\langle d_X\rangle}/X]]\!]\,\eta\varepsilon \quad \square
\end{aligned}$$

Lemma 12. *Let ϕ, ψ and ρ be arbitrary predicate formulae. Then we have the following correspondence:* $(\phi[\psi_{\langle d_X\rangle}/X])[\rho_{\langle d_X\rangle}/X] \leftrightarrow \phi[\psi[\rho_{\langle d_X\rangle}/X]_{\langle d_X\rangle}/X]$.

Proof. Let ϕ, ψ and ρ be arbitrary predicate formulae. Let θ be an arbitrary predicate environment and ε an arbitrary data environment. We show the following equivalence:

$$[\![(\phi[\psi_{\langle d_X\rangle}/X])[\rho_{\langle d_X\rangle}/X]]\!]\ \theta\varepsilon = [\![\phi[\psi[\rho_{\langle d_X\rangle}/X]/X_{\langle d_X\rangle}]]\!]\ \theta\varepsilon$$

Every non-annotated step in the derivation below utilises Property 9 once:

$$\begin{aligned}
&[\![(\phi[\psi_{\langle d_X\rangle}/X])[\rho_{\langle d_X\rangle}/X]]\!]\ \theta\varepsilon\\
=\ &\\
&[\![\phi[\psi_{\langle d_X\rangle}/X]]\!]\ \theta[\,[\![\rho_{\langle d_X\rangle}]\!]\,\theta\varepsilon/X]\varepsilon\\
=\ &\\
&[\![\phi]\!]\ (\theta[\,[\![\rho_{\langle d_X\rangle}]\!]\,\theta\varepsilon/X])[\,[\![\psi_{\langle d_X\rangle}]\!]\,\theta[\,[\![\rho_{\langle d_X\rangle}]\!]\,\theta\varepsilon/X]\varepsilon/X]\varepsilon\\
=\ &\\
&[\![\phi]\!]\ (\theta[\,[\![\rho_{\langle d_X\rangle}]\!]\,\theta\varepsilon/X])[\,[\![\psi[\rho_{\langle d_X\rangle}/X]]\!]\ \theta\varepsilon/X]\varepsilon\\
=\ &\{\text{For arbitrary functions } f \text{ and } g, \text{ we have } (\theta[f/X])[g/X] = \theta[g/X]\}\\
&[\![\phi]\!]\,\theta[\,[\![\psi[\rho_{\langle d_X\rangle}/X]_{\langle d_X\rangle}]\!]\ \theta\varepsilon/X]\varepsilon\\
=\ &\\
&[\![\phi[\psi[\rho_{\langle d_X\rangle}/X]/X_{\langle d_X\rangle}]]\!]\ \theta\varepsilon \quad \square
\end{aligned}$$

Lemma 13. *Let ϕ, ψ, ρ be arbitrary predicate formulae. Whenever $X \notin \mathrm{occ}(\rho)$ and $X \neq Y$, then* $(\phi[\psi_{\langle d_X\rangle}/X])[\rho_{\langle d_Y\rangle}/Y] \leftrightarrow (\phi[\rho_{\langle d_Y\rangle}/Y])[\psi[\rho_{\langle d_Y\rangle}/Y]_{\langle d_X\rangle}/X]$.

Proof. Let ϕ, ψ, ρ be arbitrary predicate formulae. Assume $X \notin \mathrm{occ}(\rho)$ and $X \neq Y$. Let θ, ε be arbitrary environments. We show the following equivalence:

$$[\![(\phi[\psi/X])[\rho/Y]]\!]\ \theta\varepsilon = [\![(\phi[\rho/Y])[\psi[\rho/Y]/X]]\!]\ \theta\varepsilon$$

Let θ be an arbitrary predicate environment and let ε be an arbitrary data environment. Again, every non-annotated step in the derivation below utilises Property 9 exactly once.

$$\begin{aligned}
&[\![(\phi[\psi_{\langle d_X\rangle}/X])[\rho_{\langle d_Y\rangle}/Y]]\!]\ \theta\varepsilon\\
=\ &\\
&[\![\phi[\psi_{\langle d_X\rangle}/X]]\!]\ \theta[\,[\![\rho_{\langle d_Y\rangle}]\!]\ \theta\varepsilon/Y]\varepsilon\\
=\ &\\
&[\![\phi]\!]\ (\theta[\,[\![\rho_{\langle d_Y\rangle}]\!]\ \theta\varepsilon/Y])[\,[\![\psi_{\langle d_X\rangle}]\!]\ (\theta[\,[\![\rho_{\langle d_Y\rangle}]\!]\ \theta\varepsilon/Y])\varepsilon/X]\varepsilon\\
=\ &\\
&[\![\phi]\!]\ (\theta[\,[\![\rho_{\langle d_Y\rangle}]\!]\ \theta\varepsilon/Y])[\,[\![\psi[\rho_{\langle d_Y\rangle}/Y]_{\langle d_X\rangle}]\!]\ \theta\varepsilon/X]\varepsilon\\
=\ &\{X \neq Y, \text{so for all functions } f, g, (\theta[f/X])[g/Y] = (\theta[g/Y])[f/X]\ \}\\
&[\![\phi]\!]\ (\theta[\,[\![\psi[\rho_{\langle d_Y\rangle}/Y]_{\langle d_X\rangle}]\!]\ \theta\varepsilon/X])[\,[\![\rho_{\langle d_Y\rangle}]\!]\ \theta\varepsilon/Y]\varepsilon\\
=\ &\{X \notin \mathrm{occ}(\rho), \text{so we have } [\![\rho_{\langle d_Y\rangle}]\!]\ \theta\varepsilon = [\![\rho_{\langle d_Y\rangle}]\!]\ \theta[\,[\![\psi[\rho_{\langle d_Y\rangle}/Y]_{\langle d_X\rangle}]\!]\ \theta\varepsilon/X]\varepsilon\ \}\\
&[\![\phi]\!]\ (\theta[\,[\![\psi[\rho_{\langle d_Y\rangle}/Y]_{\langle d_X\rangle}]\!]\ \theta\varepsilon/X])[\,[\![\rho_{\langle d_Y\rangle}]\!]\ (\theta[\,[\![\psi[\rho_{\langle d_Y\rangle}/Y]_{\langle d_X\rangle}]\!]\ \theta\varepsilon/X])/Y]\varepsilon\\
=\ &\\
&[\![\phi[\rho_{\langle d_Y\rangle}/Y]]\!]\ \theta[\,[\![\psi[\rho_{\langle d_Y\rangle}/Y]_{\langle d_X\rangle}]\!]\ \theta\varepsilon/X]\varepsilon\\
=\ &\\
&[\![(\phi[\rho_{\langle d_Y\rangle}/Y])[\psi[\rho_{\langle d_Y\rangle}/Y]_{\langle d_X\rangle}/X]]\!]\ \theta\varepsilon \quad \square
\end{aligned}$$

The interplay between the equivalence $\leftrightarrow$ on predicates and the notion of syntactic substitutions is quite delicate. For instance, one may believe that the following implication holds:

$$\begin{array}{ll} & \phi[\rho_{\langle d_X\rangle}/X] \leftrightarrow \phi[\psi_{\langle d_X\rangle}/X] \\ \text{implies} & \phi[(\rho \wedge X(d_X))_{\langle d_X\rangle}/X] \leftrightarrow \phi[(\psi \wedge X(d_X))_{\langle d_X\rangle}/X] \end{array}$$

However, the following example shows that this consequence is invalid:

Example 14. Let X be a Boolean sorted predicate variable and d_X a Boolean data variable. Assume $\phi = X(\top) \vee X(\bot)$. Take $\rho := d_X$ and $\psi := \neg d_X$. Clearly, $\phi[\rho_{\langle d_X\rangle}/X] \leftrightarrow \top \leftrightarrow \phi[\psi_{\langle d_X\rangle}/X]$. However, $\phi[(\rho \wedge X(d_X))_{\langle d_X\rangle}/X] \leftrightarrow X(\top) \not\leftrightarrow \phi[(\psi \wedge X(d_X))_{\langle d_X\rangle}/X]$. □

In many cases, we wish to perform a series of substitutions, rather than a single substitution, see e.g. Lemma 13. Writing down the entire sequence of substitutions in case all substitutions are similar is quite involved; we therefore generalise single syntactic substitutions $\phi[\psi_{\langle d_X\rangle}/X]$ to finite sequences of substitutions of the form $\phi[\psi_{1\langle d_{X_1}\rangle}/X_1][\psi_{2\langle d_{X_2}\rangle}/X_2]\ldots[\phi_{n\langle d_{X_n}\rangle}/X_n]$, where all predicate formulae ψ_i are similar, as follows:

Definition 15. Let $V = \langle X_1, \ldots, X_n\rangle$ be a vector of predicate variables and let $\phi_i\,(i = 1\ldots n)$ be arbitrary predicate formulae. The *consecutive substitution* $\phi\left[_{X_i\in V}\phi_{i\langle d_{X_i}\rangle}/X_i\right]$ for predicate formula ϕ is defined as follows:

$$\begin{cases} \phi\left[_{X_i\in\langle\rangle}\phi_{i\langle d_{X_i}\rangle}/X_i\right] = \phi \\ \phi\left[_{X_i\in\langle X_1,\ldots,X_n\rangle}\phi_{i\langle d_{X_i}\rangle}/X_i\right] = (\phi[\phi_{1\langle d_{X_1}\rangle}/X_1])\left[_{X_i\in\langle X_2,\ldots,X_n\rangle}\phi_{i\langle d_{X_i}\rangle}/X_i\right] \end{cases}$$

In case only variable X_i occurs in ϕ_i for all i and all variables in $\langle X_1, \ldots, X_n\rangle$ are distinct, the consecutive substitution $\phi\left[_{X_i\in\langle X_1,\ldots,X_n\rangle}\phi_{i\langle d_{X_i}\rangle}/X_i\right]$ yields the same for all permutations of vector $\langle X_1, \ldots, X_n\rangle$, i.e. it behaves as a simultaneous substitution. This is expressed by the following lemma.

Lemma 16. *Let $X_1, \ldots, X_n$ be distinct predicate variables, and let ϕ_i, for $1 \le i \le n$, be predicate formulae for which at most variable X_i occurs in ϕ_i. Then for all permutations $\pi : \{1, \ldots, n\} \to \{1, \ldots, n\}$:*

$$\phi\left[_{X_i\in\langle X_1,\ldots,X_n\rangle}\phi_{i\langle d_{X_i}\rangle}/X_i\right] \leftrightarrow \phi\left[_{X_i\in\langle X_{\pi(1)},\ldots,X_{\pi(n)}\rangle}\phi_{i\langle d_{X_i}\rangle}/X_i\right]$$

Proof. Follows by an induction on the length of the vector $\langle X_1, \ldots, X_n\rangle$, and the observation that $\phi_i[\phi_{j\langle d_{X_j}\rangle}/X_j] \leftrightarrow \phi_i$ for all $i \neq j$. □

In case the consecutive substitution behaves as a simultaneous substitution, we allow abuse of notation by writing $\phi\left[_{X_i\in\{X_1,\ldots,X_n\}}\phi_{i\langle d_{X_i}\rangle}/X_i\right]$.

2.3. Parameterised Boolean Equation Systems

A Parameterised Boolean Equation System (PBES) is a finite sequence of equations of the form

$$\sigma X(d_X{:}D_X) = \phi$$

ϕ is a predicate formula in which the variable d_X is considered bound in the equation for X; σ denotes either the least (μ) or the greatest (ν) fixed point. We denote the empty PBES by ϵ.

In the remainder of this paper, we abbreviate the term Parameterised Boolean Equation System to *equation system*. We say an equation system is *closed* whenever every predicate variable occurring at the right-hand side of some equation occurs at the left-hand side of some equation. An equation system is *open* if it is not closed. For a given equation system $\mathcal{E}$, the *defined variables* are the predicate variables occurring in the left-hand side of the equations of $\mathcal{E}$; these are collected in the set $\mathsf{bnd}(\mathcal{E})$. An equation is a *defining* equation for a predicate variable X if X is the equation's defined variable. The predicate variables occurring in the predicate formulae of the equations of an equation system $\mathcal{E}$ are collected in the set $\mathsf{occ}(\mathcal{E})$. The *solution* to an equation system is defined in the context of a predicate environment, and assigns *functions* to every defined variable:

Definition 17. Given a predicate environment θ and an equation system $\mathcal{E}$, the *solution* $[\![\mathcal{E}]\!]\theta\varepsilon$ is an environment that is defined as follows:

$$\begin{array}{l} [\![\epsilon]\!]\theta\varepsilon = \theta \\ [\![(\sigma X(d_X{:}D_X) = \phi)\mathcal{E}]\!]\theta\varepsilon = [\![\mathcal{E}]\!]\left(\theta\left[\sigma\mathcal{X}{\in}[D_X \to B].\,[\![\phi_{\langle d_X\rangle}]\!]\,([\![\mathcal{E}]\!]\theta[\mathcal{X}/X])\varepsilon/X\right]\right)\varepsilon \end{array}$$

Note that the fixed points are taken over the complete lattice of functions $([D_X \to B], \sqsubseteq)$ for (possibly infinite) data sets D_X, where $f \sqsubseteq g$ is defined as the point-wise ordering: $f \sqsubseteq g$ iff for all $v \in D_X$: $f(v)$ implies $g(v)$. The predicate transformer associated to a predicate function $[\![\phi_{\langle d_X \rangle}]\!]\,\theta\varepsilon$, denoted

$$\lambda \mathcal{X} \in [D_X \to B].\ [\![\phi_{\langle d_X \rangle}]\!]\,\theta[\mathcal{X}/X]\varepsilon$$

is a monotone operator [14,15,11]. The existence of the (extremal) fixed points of this operator in the lattice $([D_X \to B], \sqsubseteq)$ follows immediately from Tarski's fixed point Theorem [27]. A standard, constructive technique for computing a fixed point is by means of a transfinite approximation over the ordinals (see, e.g. [19]).

Definition 18. Let $(D, \leq)$ be a complete lattice with $\top$ and $\bot$ as top and bottom elements. Let $f{:}D \to D$ be a monotone function. Then $\sigma^{\alpha} X.f(X)$ is an approximant term, where α is an ordinal. The approximant terms are defined by transfinite induction, where λ is a limit ordinal:

$$\begin{aligned} \sigma^{0} X.f(X) &= \top \text{ if } \sigma = \nu \text{ and } \bot \text{ else} \\ \sigma^{\alpha+1} X.f(X) &= f(\sigma^{\alpha} X.f(X)) \\ \sigma^{\lambda} X.f(X) &= \bigwedge_{\alpha<\lambda} \sigma^{\alpha} X.f(X) \text{ if } \sigma = \nu \text{ and } \bigvee_{\alpha<\lambda} \sigma^{\alpha} X.f(X) \text{ else} \end{aligned}$$

The solution of an equation system is sensitive to the ordering of the equations. For instance, the equation system $(\mu X = Y)(\nu Y = X)$ has as solution $\bot$ for X and Y, whereas the equation system $(\nu Y = X)(\mu X = Y)$ has as solution $\top$ for X and Y. However, it is known that applying any of the following three basic transformations, viz. migration, substitution and unfolding, does not affect the solution of an equation system [15,28]:

Lemma 19. *Let $\mathcal{E}_0, \mathcal{E}_1, \mathcal{E}_2$ be arbitrary equation systems and let X, Y be predicate variables with $X, Y \notin \mathsf{bnd}(\mathcal{E}_i)$ for $i = 0..2$. Then:*

- *(Migration) Let ϕ be a simple predicate formula. Let*

$$\begin{aligned} \mathcal{E} &:\equiv \mathcal{E}_0\ (\sigma X(d_X{:}D_X) = \phi)\ \mathcal{E}_1\ \mathcal{E}_2 \quad \textit{and} \\ \mathcal{F} &:\equiv \mathcal{E}_0\ \mathcal{E}_1\ (\sigma X(d_X{:}D_X) = \phi)\ \mathcal{E}_2 \end{aligned}$$

- *(Unfolding) Let ϕ be an arbitrary predicate formula. Let*

$$\begin{aligned} \mathcal{E} &:\equiv \mathcal{E}_0\ (\sigma X(d_X{:}D_X) = \phi)\ \mathcal{E}_1 \quad \textit{and} \\ \mathcal{F} &:\equiv \mathcal{E}_0\ (\sigma X(d_X{:}D_X) = \phi[\phi_{\langle d_X \rangle}/X])\ \mathcal{E}_1 \end{aligned}$$

- *(Substitution) Let ϕ and ψ be arbitrary predicate formulae. Let*

$$\begin{aligned} \mathcal{E} &:\equiv \mathcal{E}_0\ (\sigma X(d_X{:}D_X) = \phi)\ \mathcal{E}_1\ (\sigma' Y(d_Y{:}D_Y) = \psi)\ \mathcal{E}_2 \quad \textit{and} \\ \mathcal{F} &:\equiv \mathcal{E}_0\ (\sigma X(d_X{:}D_X) = \phi[\psi_{\langle d_Y \rangle}/Y])\ \mathcal{E}_1\ (\sigma' Y(d_Y{:}D_Y) = \psi)\ \mathcal{E}_2 \end{aligned}$$

In all three cases, $\mathcal{E}$ and $\mathcal{F}$ have the same solution, regardless of the predicate environments and data environments that are used, see [15,28].

Using migration and substitution, all equation systems can be solved, provided that one has the techniques and tools to eliminate a predicate variable from its defining equation. The strategy underlying the solution method is reminiscent of Gauß Elimination in Linear Algebra. For a detailed account for PBESs, see [15]; for the subclass of *Boolean Equation Systems*, see [19].

For the sake of completeness, we recall the solution technique of *symbolic approximation* [14,15], as this technique is used frequently in the examples throughout this paper. Let $\phi[\psi_{\langle d_X \rangle}/X]^k$ be defined as:

$$\begin{cases} \psi & \text{if } k = 0 \\ \phi[\phi[\psi_{\langle d_X \rangle}/X]^{k-1}{}_{\langle d_X \rangle}/X] & \text{if } k > 0 \end{cases}$$

Proposition 20 (*See [15]*). *Let ϕ be a predicate formula, $k{:}N$ be a natural number and $\mathcal{E}_0, \mathcal{E}_1$ be equation systems. Let η, ε be arbitrary environments. Then*

1. *If $\phi[\top_{\langle d_X \rangle}/X]^k \leftrightarrow \phi[\top_{\langle d_X \rangle}/X]^{k+1}$ then*
   $[\![\mathcal{E}_0\ (\nu X(d_X{:}D_X) = \phi)\ \mathcal{E}_1]\!]\,\eta\varepsilon = [\![\mathcal{E}_0\ (\nu X(d_X{:}D_X) = \phi[\top_{\langle d_X \rangle}/X]^k)\ \mathcal{E}_1]\!]\,\eta\varepsilon$
2. *If $\phi[\bot_{\langle d_X \rangle}/X]^k \leftrightarrow \phi[\bot_{\langle d_X \rangle}/X]^{k+1}$ then*
   $[\![\mathcal{E}_0\ (\mu X(d_X{:}D_X) = \phi)\ \mathcal{E}_1]\!]\,\eta\varepsilon = [\![\mathcal{E}_0\ (\mu X(d_X{:}D_X) = \phi[\bot_{\langle d_X \rangle}/X]^k)\ \mathcal{E}_1]\!]\,\eta\varepsilon$. □

3. Invariants

Throughout the literature, (inductive) invariants play an important role in the analysis of systems that deal with iteration and recursion. Invariants for equation systems first appeared in [15]. The definition of an invariant, as stated in [15] is as follows:

Definition 21. Let $(\sigma X(d_X{:}D_X) = \phi)$ be an equation and let I be a simple predicate formula. Then I is an invariant of X iff

$$I \wedge \phi \;\leftrightarrow\; (I \wedge \phi)[(I \wedge X(d_X))_{\langle d_X\rangle}/X]$$

Observe that the invariance condition only concerns a transfer property on equation systems; an initialisation criterion is not applicable in our setting, since equation systems have no notion of "initial state". However, an analogue to the initialisation property is addressed in Theorem 35 and its derived corollaries in this paper (see Section 4), and Theorems 40 and 42 of [15], of which we repeat Theorem 42 for the sake of completeness:

Theorem 22 (*See [15]*). *Let $(\sigma X(d_X{:}D_X) = \phi)$ be an equation and let I be an invariant of X. Assume that:*

(1) *for all equation systems $\mathcal{E}$ and environments η, ε and χ such that $X \notin \mathrm{occ}(\chi)$:*

$$[\![(\sigma X(d_X{:}D_X) = I \wedge \phi)\ \mathcal{E}]\!]\,\eta\varepsilon = [\![(\sigma X(d_X{:}D_X) = \chi)\ \mathcal{E}]\!]\,\eta\varepsilon$$

(2) *for the predicate formula ψ we have $\psi \leftrightarrow \psi[I \wedge X(d_X)_{\langle d_X\rangle}/X]$*

Then for all equation systems $\mathcal{E}_0$, $\mathcal{E}_1$ and all environments η, ε:

$$\begin{aligned} & [\![(\sigma' Y(d_Y{:}D_Y) = \psi)\ \mathcal{E}_0(\sigma X(d_X{:}D_X) = \phi)\ \mathcal{E}_1]\!]\,\eta\varepsilon \\ = \;& [\![(\sigma' Y(d_Y{:}D_Y) = \psi[\chi_{\langle d_X\rangle}/X])\ \mathcal{E}_0(\sigma X(d_X{:}D_X) = \phi)\ \mathcal{E}_1]\!]\,\eta\varepsilon \quad \square \end{aligned}$$

Theorem 22 states that if one can show that $\psi \leftrightarrow \psi[(I \wedge X(d_X))_{\langle d_X\rangle}/X]$ (the analogue to the initialisation criterion for an invariant), and χ is the solution of X's equation strengthened with I, then it suffices to solve Y using χ for X rather than X's original solution. However, a computation of χ cannot take advantage of PBES manipulations when X's equation is *open*. Such equations arise when encoding process equivalences [4] and model checking problems [20,14]. A second issue is that invariants may "break" as a result of a substitution:

Example 23. Consider the following (constructed) closed equation system:

$$\begin{aligned} &(\mu X(n{:}N) = n \geq 2 \wedge Y(n)) \\ &(\mu Y(n{:}N) = Z(n) \vee Y(n+1)) \\ &(\mu Z(n{:}N) = n < 2 \vee Y(n-1)) \end{aligned} \tag{1}$$

The simple predicate formula $n \geq 2$ is an invariant for equation Y in equation system (1): $n \geq 2 \wedge (Z(n) \vee Y(n+1)) \;\leftrightarrow\; n \geq 2 \wedge (Z(n) \vee (n+1 \geq 2 \wedge Y(n+1)))$. However, substituting $n < 2 \vee Y(n-1)$ for Z in the equation of Y in system (1) yields the equation system of (2):

$$\begin{aligned} &(\mu X(n{:}N) = n \geq 2 \wedge Y(n)) \\ &(\mu Y(n{:}N) = n < 2 \vee Y(n-1) \vee Y(n+1)) \\ &(\mu Z(n{:}N) = n < 2 \vee Y(n-1)) \end{aligned} \tag{2}$$

The invariant $n \geq 2$ of Y in (1) fails to be an invariant for Y in (2). Worse still, computing the solution to Y without relying on the equation for Z leads to an awkward approximation process that does not terminate; one has to resort to using a pattern to obtain the solution to equation Y of (1):

$$(\mu Y(n{:}N) = n \geq 2 \wedge \exists i{:}N.\, Z(n+i))$$

Using this solution for Y in the equation for X in (1), and solving the resulting equation system leads to the solution $\lambda v{:}N.\ v \geq 2$ for X and $\lambda v{:}N.\ \top$ for Y and Z. A weakness of Theorem 22 is that in solving the invariant-strengthened equation for Y, one cannot employ knowledge about the equation system at hand as this is prevented by the strict conditions of Theorem 22. Weakening these conditions to incorporate information about the actual equation system is impossible without affecting correctness: solving, e.g., the invariant-strengthened version for Y of (2) leads to the solution $\lambda v{:}N.\ \bot$ for X. Theorem 40 of [15] is ungainly as it even introduces extra equations. $\square$

Example 23 shows that identified invariants (cf. [15]) fail to remain invariants when substitution is exercised on the equation system, and, more importantly, that Theorem 22 cannot employ PBES manipulations for simplifying the invariant-strengthened equation.

As we demonstrate in this paper, both issues can be remedied by using a slightly stronger invariance criterion, taking all predicate variables of an equation system into account. This naturally leads to a notion of *global invariance*; in contrast, we refer to the type of invariance defined in Definition 21 as *local invariance*.

To facilitate notation, we introduce the following terminology: a function $f{:}V \to \mathsf{Pred}$, with $V \subseteq \mathcal{P}$, is called *simple* iff for all $X \in V$, the predicate $f(X)$ is simple. Note that the notation $f(X)$ is *meta-notation*, i.e. it is not affected by e.g. syntactic substitutions: $f(X)[\psi_{\langle d_X\rangle}/X]$ remains $f(X)$, since $f(X)$ is simple.

Definition 24. The simple function $f{:}V \to \mathsf{Pred}$ is said to be a *global invariant* for an equation system $\mathcal{E}$ iff $V \supseteq \mathsf{bnd}(\mathcal{E})$ and for each $(\sigma X(d_X{:}D_X) = \phi)$ occurring in $\mathcal{E}$, we have:

$$f(X) \wedge \phi \;\leftrightarrow\; (f(X) \wedge \phi)\left[_{X_i \in V}(f(X_i) \wedge X_i(d_{X_i}))_{\langle d_{X_i}\rangle}/X_i\right] \tag{3}$$

The following proposition relates local and global invariants, and is instrumental in proving the main theorem of the next section.

Proposition 25. *Let $f{:}V \to \mathsf{Pred}$ be a global invariant for an equation system $\mathcal{E}$ and let $W \subseteq V$. Then for every equation $(\sigma X(d_X{:}D_X) = \phi)$ in $\mathcal{E}$, we have:*

$$f(X) \wedge \phi \;\leftrightarrow\; (f(X) \wedge \phi)\left[_{X_i \in W}(f(X_i) \wedge X_i(d_{X_i}))_{\langle d_{X_i}\rangle}/X_i\right] \tag{4}$$

Proof. Let $f : V \to \mathsf{Pred}$ be a global invariant for $\mathcal{E}$. Let $(\sigma X(d_X : D_X) = \phi)$ be an arbitrary equation in $\mathcal{E}$. We prove the following property for all $W \subseteq V$:

$$f(X) \wedge \phi \;\leftrightarrow\; (f(X) \wedge \phi)\left[_{X_i \in W}(f(X_i) \wedge X_i(d_{X_i}))_{\langle d_{X_i}\rangle}/X_i\right]$$

We use induction on the size of the set W.

(1) *Base case*: $W = \emptyset$. Then $(f(X) \wedge \phi)\left[_{X_i \in W}(f(X_i) \wedge X_i(d_{X_i}))_{\langle d_{X_i}\rangle}/X_i\right]$ is defined as $f(X) \wedge \phi$. By reflexivity of $\leftrightarrow$, we find that the property holds for $W = \emptyset$.

(2) *Induction*: assume that for $W \subset V$ we have:

$$f(X) \wedge \phi \;\leftrightarrow\; (f(X) \wedge \phi)\left[_{X_i \in W}(f(X_i) \wedge X_i(d_{X_i}))_{\langle d_{X_i}\rangle}/X_i\right] \tag{IH}$$

Assume that $X_j \notin W$. Then:

$$\begin{aligned}
&(f(X) \wedge \phi)\left[_{X_i \in W \cup \{X_j\}}(f(X_i) \wedge X_i(d_{X_i}))_{\langle d_{X_i}\rangle}/X_i\right]\\
\leftrightarrow\ &\{\text{Property of consecutive substitution }\}\\
&\Big((f(X) \wedge \phi)\left[_{X_i \in W}(f(X_i) \wedge X_i(d_{X_i}))_{\langle d_{X_i}\rangle}/X_i\right]\Big)\\
&\qquad [(f(X_j) \wedge X_j(d_{X_j}))_{\langle d_{X_j}\rangle}/X_j]\\
\leftrightarrow\ &\{\text{Lemma 11 and (IH)}\}\\
&((f(X) \wedge \phi))\,[(f(X_j) \wedge X_j(d_{X_j}))_{\langle d_{X_j}\rangle}/X_j]\\
\leftrightarrow\ &\{\text{Lemma 11 and } f \text{ is a global invariant }\}\\
&\Big((f(X) \wedge \phi)\left[_{X_i \in V}(f(X_i) \wedge X_i(d_{X_i}))_{\langle d_{X_i}\rangle}/X_i\right]\Big)\\
&\qquad [(f(X_j) \wedge X_j(d_{X_j}))_{\langle d_{X_j}\rangle}/X_j]\\
\leftrightarrow\ &\{\text{Property of consecutive substitution}\}\\
&\Big((f(X) \wedge \phi)\left[_{X_i \in V \setminus \{X_j\}}(f(X_i) \wedge X_i(d_{X_i}))_{\langle d_{X_i}\rangle}/X_i\right]\\
&\qquad [(f(X_j) \wedge X_j(d_{X_j}))_{\langle d_{X_j}\rangle}/X_j]\Big)\ [(f(X_j) \wedge X_j(d_{X_j}))_{\langle d_{X_j}\rangle}/X_j]\\
\leftrightarrow\ &\{\text{Lemma 12}\}\\
&\Big((f(X) \wedge \phi)\left[_{X_i \in V \setminus \{X_j\}}(f(X_i) \wedge X_i(d_{X_i}))_{\langle d_{X_i}\rangle}/X_i\right]\\
&\qquad [(f(X_j) \wedge f(X_j) \wedge X_j(d_{X_j}))_{\langle d_{X_j}\rangle}/X_j]\Big)\\
\leftrightarrow\ &\{\text{Idempotence of } \wedge\}\\
&\Big((f(X) \wedge \phi)\left[_{X_i \in V}(f(X_i) \wedge X_i(d_{X_i}))_{\langle d_{X_i}\rangle}/X_i\right]\\
&\qquad [(f(X_j) \wedge X_j(d_{X_j}))_{\langle d_{X_j}\rangle}/X_j]\Big)\\
\leftrightarrow\ &\{\text{Property of consecutive substitution; } f \text{ is a global invariant}\}\\
&f(X) \wedge \phi \quad \square
\end{aligned}$$

Corollary 26. *For any global invariant f for an equation system $\mathcal{E}$, all predicate formulae $f(X)$ for $X \in \mathsf{bnd}(\mathcal{E})$ are local invariants.*

Remark 27. The above corollary firmly links the notions of local invariants to global invariants. However, one should be aware that the reverse of this corollary does not hold: if for all $X \in \mathsf{bnd}(\mathcal{E})$, we have a predicate formula $f(X)$ that is a local invariant for X in $\mathcal{E}$, then f is *not* necessarily a global invariant. This is illustrated by the following equation system: $(\nu X(n{:}N) = Y(n-1))(\mu Y(n{:}N) = X(n+1))$. The simple predicate $n \geq 5$ is a local invariant for both X and Y, but the simple function $f(X) = f(Y) = (n \geq 5)$ is *not* a global invariant.

Finding useful invariants can be a challenging task. The following property gives a sufficient condition for a simple function f to be a global invariant. We first define the set of *predicate variable instantiations* occurring in a formula ϕ:

$$\begin{array}{rcrcl} \mathsf{pvi}(b) & = & \emptyset & \quad \mathsf{pvi}(X(e)) & = & \{X(e)\} \\ \mathsf{pvi}(\forall d{:}D.\ \phi) & = & \mathsf{pvi}(\phi) & \mathsf{pvi}(\phi_1 \wedge \phi_2) & = & \mathsf{pvi}(\phi_1) \cup \mathsf{pvi}(\phi_2) \\ \mathsf{pvi}(\exists d{:}D.\ \phi) & = & \mathsf{pvi}(\phi) & \mathsf{pvi}(\phi_1 \vee \phi_2) & = & \mathsf{pvi}(\phi_1) \cup \mathsf{pvi}(\phi_2) \end{array}$$

Property 28. *Let $\mathcal{E}$ be a closed equation system. Let $f{:}\mathsf{bnd}(\mathcal{E}) \to \mathsf{Pred}$ be a simple function such that for every equation $(\sigma X(d_X{:}D_X) = \phi)$ in $\mathcal{E}$ we have:*

$$f(X) \to \bigwedge_{Y(e) \in \mathsf{pvi}(\phi)} (f(Y))[e/d_Y]$$

Then f is a global invariant for $\mathcal{E}$.

Proof. Let us consider an equation $(\sigma X(d_X{:}D_X) = \phi)$ for which the implication above holds. As a consequence, for any subformula ψ of ϕ it holds that $f(X) \to \bigwedge_{Y(e) \in \mathsf{pvi}(\psi)} (f(Y))[e/d_Y]$. Using an induction on the structure of the subformulae ψ of ϕ, we prove that the following equivalence holds, for $V = \mathsf{bnd}(\mathcal{E})$:

$$f(X) \wedge \psi \leftrightarrow (f(X) \wedge \psi) \left[_{Z \in V} (f(Z) \wedge Z(d_Z))_{\langle d_Z \rangle} / Z\right].$$

We first address the base cases:

- Case $\psi = b$. By definition of syntactic substitution, we immediately obtain $f(X) \wedge b \leftrightarrow (f(X) \wedge b) \left[_{Z \in V} (f(Z) \wedge Z(d_Z))_{\langle d_Z \rangle} / Z\right]$.
- Case $\psi = Y(e)$. We reason as follows:

$$\begin{array}{ll} & (f(X) \wedge Y(e)) \left[_{Z \in V} (f(Z) \wedge Z(d_Z))_{\langle d_Z \rangle} / Z\right] \\ \leftrightarrow & \{\text{Definition of syntactic substitution and } f(X) \text{ simple}\} \\ & f(X) \wedge Y(e) \left[_{Z \in V} (f(Z) \wedge Z(d_Z))_{\langle d_Z \rangle} / Z\right] \\ \leftrightarrow & \{\text{Definition of syntactic substitution}\} \\ & f(X) \wedge (f(Y)[e/d_Y]) \wedge Y(e) \\ \leftrightarrow & \{f(X) \to f(Y)[e/d_Y], \text{therefore } f(X) \wedge (f(Y)[e/d_Y]) \leftrightarrow f(X)\} \\ & f(X) \wedge Y(e) \end{array}$$

We assume the following induction hypothesis: for arbitrary subformulae ψ_i of ϕ, we have:

$$f(X) \wedge \psi_i \leftrightarrow (f(X) \wedge \psi_i) \left[_{Z \in V} (f(Z) \wedge Z(d_Z))_{\langle d_Z \rangle} / Z\right] \tag{IH}$$

- Case $\psi = \psi_1 \wedge \psi_2$. Then:

$$\begin{array}{ll} & (f(X) \wedge \psi_1 \wedge \psi_2) \left[_{Z \in V} (f(Z) \wedge Z(d_Z))_{\langle d_Z \rangle} / Z\right] \\ \leftrightarrow & \{f(X) = f(X) \wedge f(X), \text{definition of syntactic substitution}\} \\ & (f(X) \wedge \psi_1) \left[_{Z \in V} (f(Z) \wedge Z(d_Z))_{\langle d_Z \rangle} / Z\right] \\ & \wedge\ (f(X) \wedge \psi_2) \left[_{Z \in V} (f(Z) \wedge Z(d_Z))_{\langle d_Z \rangle} / Z\right] \\ \leftrightarrow & \{\text{Induction hypothesis}\} \\ & (f(X) \wedge \psi_1) \wedge (f(X) \wedge \psi_2) \\ \leftrightarrow & \{\psi_1 \wedge \psi_2 = \psi\} \\ & f(X) \wedge \psi \end{array}$$

The case for $\psi = \psi_1 \vee \psi_2$ is similar.

- Case $\psi = \forall e{:}E.\ \psi_1$. Without loss of generality, we assume that e does not occur in $f(X)$. Suitable α-renaming can ensure this is the case.

$$
\begin{array}{ll}
 & (f(X) \wedge \forall e{:}E.\ \psi_1)\ \left[_{Z\in V}(f(Z) \wedge Z(d_Z))_{\langle d_Z\rangle}/Z\right]\ ) \\
\leftrightarrow & \{e \text{ does not occur in } f(X)\} \\
 & \forall e{:}E.\ (f(X) \wedge \psi_1)\ \left[_{Z\in V}(f(Z) \wedge Z(d_Z))_{\langle d_Z\rangle}/Z\right] \\
\leftrightarrow & \{\text{Induction hypothesis}\} \\
 & \forall e{:}E.\ (f(X) \wedge \psi_1) \\
\leftrightarrow & \{e \text{ does not occur in } f(X)\} \\
 & f(X) \wedge \forall e{:}E.\ \psi_1
\end{array}
$$

  The case for $\psi = \exists e{:}E.\ \psi_1$ is similar. □

Note that the condition of Property 28 is not a necessary condition. For instance, the equation system given by the single (trivial) equation $(\mu X(n{:}N) = X(n{+}1) \vee \top)$ does not fulfill the condition of Property 28. Yet, all simple functions are global invariants for this equation system. With the same purpose of easing the task of invariant checking, we give one more sufficient condition for a simple function to meet the condition of the global invariant definition (Definition 24).

Property 29. *Let $(\sigma X(d{:}D) = \phi)$, with $\phi = \chi \wedge \bigwedge_{i\in I}(\psi_i \implies X_i(e_i))$, be an equation. For all i, χ and ψ_i are simple predicate formulae, $X_i \in V$, and e_i is a data term. Moreover, let $f{:}V \to \mathsf{Pred}$ be a simple function such that, for all i, $f(X) \wedge \chi \wedge \psi_i \to f(X_i)[e_i/d_{X_i}]$. Then $f(X) \wedge \phi \leftrightarrow (f(X) \wedge \phi)\ \left[_{X_i\in V}(f(X_i) \wedge X_i(d_{X_i}))_{\langle d_{X_i}\rangle}/X_i\right]$.*

Proof. Let us start with the right-hand side of the equality to prove:

$$
\begin{array}{ll}
 & (f(X) \wedge \phi)\ \left[_{X_i\in V}(f(X_i) \wedge X_i(d_{X_i}))_{\langle d_{X_i}\rangle}/X_i\right] \\
\leftrightarrow & \{\text{Expansion of } \phi, f(X) \text{ and } \chi \text{ are simple}\} \\
 & f(X) \wedge \chi \wedge \bigwedge_{i\in I}(\psi_i \implies f(X_i)[e_i/d_{X_i}] \wedge X_i(e_i)) \\
\leftrightarrow & \{\text{For any } \alpha, \beta, \gamma{:}\ (\alpha \implies \beta \wedge \gamma) \leftrightarrow (\alpha \implies \beta \wedge \alpha \implies \gamma)\} \\
 & f(X) \wedge \chi \wedge \bigwedge_{i\in I}(\psi_i \implies f(X_i)[e_i/d_{X_i}]) \wedge \bigwedge_{i\in I}(\psi_i \implies X_i(e_i)) \\
\leftrightarrow & \{\text{For any } \alpha, \beta, \gamma{:}\ (\alpha \wedge (\beta \implies \gamma)) \leftrightarrow \alpha \wedge ((\alpha \wedge \beta) \implies \gamma)\} \\
 & f(X) \wedge \chi \wedge \bigwedge_{i\in I}((f(X) \wedge \chi \wedge \psi_i) \implies f(X_i)[e_i/d_{X_i}]) \\
 & \quad \wedge \bigwedge_{i\in I}(\psi_i \implies X_i(e_i)) \\
\leftrightarrow & \{\text{For all } i, f(X) \wedge \chi \wedge \psi_i \to f(X_i)[e_i/d_{X_i}]\} \\
 & f(X) \wedge \chi \wedge \top \wedge \bigwedge_{i\in I}(\psi_i \implies X_i(e_i)) \\
 & \{\text{Definition of } \phi\} \\
\leftrightarrow & f(X) \wedge \phi \quad \square
\end{array}
$$

Invariants can be combined using logical connectives $\wedge$ and $\vee$. Let $f, g{:}V \to \mathsf{Pred}$ be arbitrary simple functions. We write $f \wedge g$ to denote the function $\lambda Z{\in}V.\ f(Z) \wedge g(Z)$. Likewise, we define $f \vee g$ as the function $\lambda Z{\in}V.\ f(Z) \vee g(Z)$.

Lemma 30. *Let ϕ be an arbitrary predicate formula and let $f, g{:}V \to \mathsf{Pred}$ be simple functions. If the following three conditions are met:*

(1) $\mathsf{occ}(\phi) \subseteq V$,
(2) $f(X) \wedge \phi \leftrightarrow (f(X) \wedge \phi)\ \left[_{Z\in V}(f(Z) \wedge Z(d_Z))_{\langle d_Z\rangle}/Z\right]$,
(3) $g(X) \wedge \phi \leftrightarrow (g(X) \wedge \phi)\ \left[_{Z\in V}(g(Z) \wedge Z(d_Z))_{\langle d_Z\rangle}/Z\right]$.

then also:

$$
\begin{array}{l}
(f \wedge g)(X) \wedge \phi \leftrightarrow ((f \wedge g)(X) \wedge \phi)\ \left[_{Z\in V}((f \wedge g)(Z) \wedge Z(d_Z))_{\langle d_Z\rangle}/Z\right] \\
\text{and } (f \vee g)(X) \wedge \phi \leftrightarrow ((f \vee g)(X) \wedge \phi)\ \left[_{Z\in V}((f \vee g)(Z) \wedge Z(d_Z))_{\langle d_Z\rangle}/Z\right]
\end{array}
$$

Proof. Let $f, g{:}V \to \mathsf{Pred}$ be arbitrary simple predicate formulae. We only consider the case for $f \wedge g$, since the case for $f \vee g$ follows the same line of reasoning. We prove the property using an induction on the structure of ϕ. We first address the base cases.

- Case $\phi = b$. By definition of syntactic substitution, we immediately obtain $(f \wedge g)(X) \wedge b \leftrightarrow ((f \wedge g)(X) \wedge b)\ \left[_{Z\in V}((f \wedge g)(Z) \wedge Z(d_Z))_{\langle d_Z\rangle}/Z\right]$.

- Case $\phi = Y(e)$, where Y is an arbitrary predicate variable. Assume the three conditions of the lemma are satisfied for ϕ. Then:

$$\begin{aligned}
& (f \wedge g)(X) \wedge Y(e) \\
\leftrightarrow\ & f(X) \wedge g(X) \wedge Y(e) \\
\leftrightarrow\ & (f(X) \wedge Y(e)) \wedge (g(X) \wedge Y(e)) \\
\leftrightarrow^{\dagger}\ & (f(X) \wedge Y(e)) \left[_{Z\in V}(f(Z) \wedge Z(d_Z))_{\langle d_Z\rangle}/Z\right] \\
& \wedge (g(X) \wedge Y(e)) \left[_{Z\in V}(g(Z) \wedge Z(d_Z))_{\langle d_Z\rangle}/Z\right] \\
\leftrightarrow^{\ddagger}\ & (f(X) \wedge f(Y)[e/d_Y] \wedge Y(e)) \wedge (g(X) \wedge g(Y)[e/d_Y] \wedge Y(e)) \\
\leftrightarrow\ & (f \wedge g)(X) \wedge (f \wedge g)(Y)[e/d_Y] \wedge Y(e) \\
\leftrightarrow\ & ((f \wedge g)(X) \wedge Y(e)) \left[_{Z\in V}((f \wedge g)(Z) \wedge Z(d_Z))_{\langle d_Z\rangle}/Z\right]
\end{aligned}$$

  where at † we used the assumptions on f and g and ‡ we applied the definition of syntactic substitution and the fact that $Y \in V$.

  We assume the following induction hypothesis: for arbitrary formula ϕ_i satisfying the three conditions of the lemma, we have:

$$(f \wedge g)(X) \wedge \phi_i \leftrightarrow ((f \wedge g)(X) \wedge \phi_i) \left[_{Z\in V}((f \wedge g)(Z) \wedge Z(d_Z))_{\langle d_Z\rangle}/Z\right] \tag{IH}$$

- Case $\phi = \phi_1 \wedge \phi_2$. Then:

$$\begin{aligned}
& (f \wedge g)(X) \wedge \phi \\
\leftrightarrow\ & (f \wedge g)(X) \wedge \phi_1 \wedge \phi_2 \\
\leftrightarrow\ & ((f \wedge g)(X) \wedge \phi_1) \wedge ((f \wedge g)(X) \wedge \phi_2) \\
\leftrightarrow^{\{(\mathrm{IH})\}}\ & ((f \wedge g)(X) \wedge \phi_1) \left[_{Z\in V}((f \wedge g)(Z) \wedge Z(d_Z))_{\langle d_Z\rangle}/Z\right] \\
& \wedge ((f \wedge g)(X) \wedge \phi_2) \left[_{Z\in V}((f \wedge g)(Z) \wedge Z(d_Z))_{\langle d_Z\rangle}/Z\right] \\
\leftrightarrow\ & ((f \wedge g)(X) \wedge \phi_1 \wedge \phi_2) \left[_{Z\in V}((f \wedge g)(Z) \wedge Z(d_Z))_{\langle d_Z\rangle}/Z\right] \\
\leftrightarrow\ & ((f \wedge g)(X) \wedge \phi) \left[_{Z\in V}((f \wedge g)(Z) \wedge Z(d_Z))_{\langle d_Z\rangle}/Z\right]
\end{aligned}$$

  The case where $\phi = \phi_1 \vee \phi_2$ is similar but uses distributivity of $\wedge$ over $\vee$ at the second step rather than idempotence of $\wedge$.
- Case $\phi = \forall e{:}E.\ \phi_1$. Without loss of generality, we assume that e does not occur in $f(X)$ and $g(X)$. This can be guaranteed by a suitable α-renaming.

$$\begin{aligned}
& (f \wedge g)(X) \wedge \phi \\
\leftrightarrow\ & (f \wedge g)(X) \wedge \forall e{:}E.\ \phi_1 \\
\leftrightarrow^{\dagger}\ & \forall e{:}E.((f \wedge g)(X) \wedge \phi_1) \\
\leftrightarrow^{\{(\mathrm{IH})\}}\ & \forall e{:}E.((f \wedge g)(X) \wedge \phi_1) \left[_{Z\in V}((f \wedge g)(Z) \wedge Z(d_Z))_{\langle d_Z\rangle}/Z\right] \\
\leftrightarrow^{\dagger}\ & ((f \wedge g)(X) \wedge \forall e{:}E.\ \phi_1) \left[_{Z\in V}((f \wedge g)(Z) \wedge Z(d_Z))_{\langle d_Z\rangle}/Z\right]
\end{aligned}$$

  where at † we used the fact that e does not occur in $(f \wedge g)(X)$. The case for $\phi = \exists e{:}E.\ \phi_1$ is similar and therefore omitted. □

Property 31. *Let $f, g{:}V \to \mathsf{Pred}$ be global invariants for an equation system $\mathcal{E}$. Then also $f \wedge g$ and $f \vee g$ are global invariants for $\mathcal{E}$.*

Proof. Follows from Lemma 30. □

4. Invariance theorem

Invariants for equation systems are useful only if they serve a purpose in computing the solution to equation systems or evaluating predicate formulae in the context of a given equation system. We next establish an exact correspondence between the solution of an equation system $\mathcal{E}$ and the equation system $\mathcal{E}'$ which is derived from $\mathcal{E}$ by strengthening it with the global invariant. Strengthening of an equation system with an invariant is achieved by an operation named Apply. First, we prove two technical lemmata that are at the basis of the correctness of the correspondence.

The first lemma, which is closely related to Lemma 39 of [15], relates the solution to an equation that is strengthened with its local invariant (derived from a global invariant) with the solution to the original equation. Note that by strengthening the right-hand side of an equation, the solution to that equation generally becomes smaller than the solution to the original equation system (see [15]), but in most cases, the exact correspondence cannot be characterised.

Lemma 32. *Let $(\sigma X(d_X{:}D_X) = \phi)$ be a possibly open equation. Let $f{:}V \to \mathsf{Pred}$ be a simple function such that*

(1) $\mathrm{occ}(\phi) \subseteq V$
(2) $f(X) \wedge \phi \leftrightarrow (f(X) \wedge \phi)[(f(X) \wedge X(d_X))_{\langle d_X \rangle}/X]$

Then for all environments η, ε:

$$\begin{array}{l} \lambda v{\in}D_X.\ [\![f(X)]\!]\,\varepsilon[v/d_X] \wedge (\sigma\mathcal{X} \in [D_X \to B].\ [\![\phi_{\langle d_X\rangle}]\!]\,\eta[\mathcal{X}/X]\varepsilon)(v) \\ = \\ \lambda v{\in}D_X.\ [\![f(X)]\!]\,\varepsilon[v/d_X] \wedge (\sigma\mathcal{X} \in [D_X \to B].\ [\![(f(X) \wedge \phi)_{\langle d_X\rangle}]\!]\,\eta[\mathcal{X}/X]\varepsilon)(v) \end{array}$$

Proof. We prove this lemma by a transfinite approximation. So, we let X_α be the α-th approximation for $\sigma\mathcal{X} \in [D_X \to B].\ [\![\phi_{\langle d_X\rangle}]\!]\,\eta[\mathcal{X}/X]\varepsilon)$ and $\overline{X}_\alpha$ be the α-th approximation for $\sigma\mathcal{X} \in [D_X \to B].\ [\![(f(X) \wedge \phi)_{\langle d_X\rangle}]\!]\,\eta[\mathcal{X}/X]\varepsilon$, where α is an ordinal, and we show that

$$\lambda v{\in}D_X.\ [\![f(X)]\!]\,\varepsilon[v/d_X] \wedge X_\alpha(v) = \lambda v{\in}D_X.\ [\![f(X)]\!]\,\varepsilon[v/d_X] \wedge \overline{X}_\alpha(v)$$

We find:

- For $\alpha = 0$, we must distinguish between $\sigma = \nu$ and $\sigma = \mu$. If $\sigma = \nu$, it holds that $X_0 = \overline{X}_0 = \lambda v \in D_X.\ \top$. For $\sigma = \mu$ we find that $X_0 = \overline{X}_0 = \lambda v \in D_X.\ \bot$. From both cases, it follows that $\lambda v{\in}D_X.\ [\![f(X)]\!]\,\varepsilon[v/d_X] \wedge X_0(v) = \lambda v{\in}D_X.\ [\![f(X)]\!]\,\varepsilon[v/d_X] \wedge \overline{X}_0(v)$
- For $\alpha = \beta + 1$ a successor ordinal, we assume the following induction hypothesis:

$$\begin{array}{l} \lambda v{\in}D_X.\,[\![f(X)]\!]\,\varepsilon[v/d_X] \wedge X_\beta(v) \\ = \\ \lambda v{\in}D_X.\,[\![f(X)]\!]\,\varepsilon[v/d_X] \wedge \overline{X}_\beta(v) \end{array} \tag{IH}$$

Next, we continue:

$$\begin{array}{ll} & \lambda v{\in}D_X.\,[\![f(X)]\!]\,\varepsilon[v/d_X] \wedge X_{\beta+1}(v) \\ = & \{\text{By definition of approximation}\} \\ & \lambda v{\in}D_X.\,[\![f(X)]\!]\,\varepsilon[v/d_X] \wedge\ [\![\phi]\!]\,\eta[X_\beta/X]\varepsilon[v/d_X] \\ = & \{\text{Semantics; } f \text{ is a simple function}\} \\ & \lambda v{\in}D_X.\,[\![(f(X) \wedge \phi)]\!]\,\eta[X_\beta/X]\varepsilon[v/d_X] \\ = & \{\text{Assumption on } f\} \\ & \lambda v{\in}D_X.\,[\![(f(X) \wedge \phi)[(f(X) \wedge X(d_X))_{\langle d_X\rangle}/X]]\!]\,\eta[X_\beta/X]\varepsilon[v/d_X] \\ = & \{\text{Property 9: syntactic vs. semantic substitution}\} \\ & \lambda v{\in}D_X.\,[\![(f(X) \wedge \phi)]\!] \\ & \qquad ((\eta[X_\beta/X])[\,[\![(f(X) \wedge X(d_X))_{\langle d_X\rangle}]\!]\,\eta[X_\beta/X]\varepsilon[v/d_X]\ /X])\varepsilon[v/d_X] \\ = & \{\text{Semantics; } f \text{ is a simple function; simplification of environment}\} \\ & \lambda v{\in}D_X.\,[\![(f(X) \wedge \phi)]\!]\,\eta[\lambda w{\in}D_X.\,[\![f(X)]\!]\,\eta\varepsilon[w/d_X] \wedge X_\beta(w)/X]\varepsilon[v/d_X] \\ = & \{\text{Application of (IH)}\} \\ & \lambda v{\in}D_X.\,[\![(f(X) \wedge \phi)]\!]\,\eta[\lambda w{\in}D_X.\,[\![f(X)]\!]\,\eta\varepsilon[w/d_X] \wedge \overline{X}_\beta(w)/X]\varepsilon[v/d_X] \\ = & \{\text{Semantics; } f \text{ is a simple function; rewriting environment } \eta\} \\ & \lambda v{\in}D_X.\,[\![(f(X) \wedge \phi)]\!] \\ & \qquad ((\eta[\overline{X}_\beta/X])[\,[\![(f(X) \wedge X(d_X))_{\langle d_X\rangle}]\!]\,\eta[\overline{X}_\beta/X]\varepsilon[v/d]\ /X])\varepsilon[v/d_X] \\ = & \{\text{Property 9: semantic vs. syntactic substitution}\} \\ & \lambda v{\in}D_X.\,[\![(f(X) \wedge \phi)[(f(X) \wedge X(d_X))_{\langle d_X\rangle}/X]]\!]\,\eta[X_\beta/X]\varepsilon[v/d_X] \\ = & \{\text{Assumption on } f\} \\ & \lambda v{\in}D_X.\,[\![(f(X) \wedge \phi)]\!]\,\eta[X_\beta/X]\varepsilon[v/d_X] \\ = & \{\text{By definition of approximation}\} \\ & \lambda v{\in}D_X.\,[\![f(X)]\!]\,\varepsilon[v/d_X] \wedge X_{\beta+1}(v) \end{array}$$

- For α a limit ordinal and $\sigma = \mu$, we find:

$$\begin{aligned}
& \lambda v{\in}D_X.\, [\![f(X)]\!]\, \varepsilon[v/d_X] \wedge X_\alpha(v) \\
=\ & \lambda v{\in}D_X.\, [\![f(X)]\!]\, \varepsilon[v/d_X] \wedge \bigvee_{\beta<\alpha} X_\beta(v) \\
=\ & \lambda v{\in}D_X.\, \bigvee_{\beta<\alpha} [\![f(X)]\!]\, \varepsilon[v/d_X] \wedge X_\beta(v) \\
\overset{\text{(IH)}}{=}\ & \lambda v{\in}D_X.\, \bigvee_{\beta<\alpha} [\![f(X)]\!]\, \varepsilon[v/d_X] \wedge \overline{X}_\beta(v) \\
=\ & \lambda v{\in}D_X.\, [\![f(X)]\!]\, \varepsilon[v/d_X] \wedge \bigvee_{\beta<\alpha} \overline{X}_\beta(v) \\
=\ & \lambda v{\in}D_X.\, [\![f(X)]\!]\, \varepsilon[v/d_X] \wedge \overline{X}_\alpha(v)
\end{aligned}$$

The case for $\sigma = \nu$ goes along the same lines. □

The lemma below allows one, under strict conditions, to change between predicate environments, when evaluating predicate formulae.

Lemma 33. *Let ϕ be an arbitrary predicate formula. Let f be a simple formula satisfying: $f(X) \wedge \phi \ \leftrightarrow\ (f(X) \wedge \phi)\left[_{X_i\in V}(f(X_i) \wedge X_i(d_{X_i}))_{\langle d_{X_i}\rangle}/X_i\right]$, where $f{:}V \to \mathsf{Pred}$ with $\mathsf{occ}(\phi) \subseteq V$. Then for all environments η_1, η_2, ε:*

$$\begin{aligned}
& \forall Y \in V:\ [\![(f(Y) \wedge Y(d_Y))]\!]\, \eta_1\varepsilon = [\![(f(Y) \wedge Y(d_Y))]\!]\, \eta_2\varepsilon \\
& \textit{implies} \\
& [\![(f(X) \wedge \phi)]\!]\, \eta_1\varepsilon = [\![(f(X) \wedge \phi)]\!]\, \eta_2\varepsilon
\end{aligned}$$

Proof. Let f, ϕ, η_1 and η_2 be as stated. Then we reason as follows:

$$\begin{aligned}
& [\![f(X) \wedge \phi]\!]\, \eta_1\varepsilon \\
=\ & \{\text{Assumption on } f\text{; definition of } \leftrightarrow\} \\
& [\![(f(X) \wedge \phi)\left[_{X_i\in V}(f(X_i) \wedge X_i(d_{X_i}))_{\langle d_{X_i}\rangle}/X_i\right]]\!]\, \eta_1\varepsilon \\
=\ & \{\text{Property 9 for every } X_i \in V\ \} \\
& [\![f(X) \wedge \phi]\!]\, \eta_1[\,[\![(f(X_1) \wedge X_1(d_{X_1}))_{\langle d_{X_1}\rangle}]\!]\, \eta_1\varepsilon/X_1] \ldots \\
& \qquad [\,[\![(f(X_n) \wedge X_n(d_{X_n}))_{\langle d_{X_n}\rangle}]\!]\, \eta_1\varepsilon/X_n] \\
=\ & \{\text{Assumption on } \eta_1, \eta_2\} \\
& [\![f(X) \wedge \phi]\!]\, \eta_2[\,[\![(f(X_1) \wedge X_1(d_{X_1}))_{\langle d_{X_1}\rangle}]\!]\, \eta_2\varepsilon/X_1] \ldots \\
& \qquad [\,[\![(f(X_n) \wedge X_n(d_{X_n}))_{\langle d_{X_n}\rangle}]\!]\, \eta_2\varepsilon/X_n] \\
=\ & \{\text{Property 9 for every } X_i \in V\ \} \\
& [\![(f(X) \wedge \phi)\left[_{X_i\in V}(f(X_i) \wedge X_i(d_{X_i}))_{\langle d_{X_i}\rangle}/X_i\right]]\!]\, \eta_2\varepsilon \\
=\ & \{\text{Assumption on } f\text{; definition of } \leftrightarrow\} \\
& [\![f(X) \wedge \phi]\!]\, \eta_2\varepsilon \quad \square
\end{aligned}$$

The operation that strengthens a given equation system $\mathcal{E}$ with its global invariant f is given by the operation Apply, which is defined below. In short, it adds, to every right-hand side of an equation for a predicate variable X, a conjunct $f(X)$.

Definition 34. Let $f{:}V \to \mathsf{Pred}$ be a global invariant for $\mathcal{E}$. The equation system $\mathsf{Apply}\ (f, \mathcal{E})$ is then defined as follows:

$$\begin{aligned}
\mathsf{Apply}\ (f, \epsilon) &= \epsilon \\
\mathsf{Apply}\ (f, (\sigma X(d_X{:}D_X) = \phi)\ \mathcal{E}_0) &= (\sigma X(d_X{:}D_X) = f(X) \wedge \phi)\ \mathsf{Apply}\ (f, \mathcal{E}_0)
\end{aligned}$$

The formal correspondence between the solution of an equation system $\mathcal{E}$ and the equation system $\mathsf{Apply}\ (f, \mathcal{E})$ is given by the following theorem.

Theorem 35. *Let $f{:}V \to \mathsf{Pred}$ be a simple function. Then, for all equation systems $\mathcal{E}$ and for all environments η_1 and η_2, if the following conditions are met:*

(1) $\mathsf{bnd}(\mathcal{E}) \cup \mathsf{occ}(\mathcal{E}) \subseteq V$ *and*
(2) *for all $X \in V$:*
  (a) $[\![f(X) \wedge X(d_X)]\!]\, \eta_1\varepsilon = [\![f(X) \wedge X(d_X)]\!]\, \eta_2\varepsilon$
  (b) $f(X) \wedge \phi \leftrightarrow (f(X) \wedge \phi)\left[_{X_i\in V}(f(X_i) \wedge X_i(d_{X_i}))_{\langle d_{X_i}\rangle}/X_i\right]$

then we have for all $X \in V$:

$$[\![f(X) \wedge X(d_X))]\!]\, (\,[\![\mathcal{E}]\!]\, \eta_1\varepsilon)\varepsilon = [\![f(X) \wedge X(d_X))]\!]\, (\,[\![\mathsf{Apply}\ (f, \mathcal{E})]\!]\, \eta_2\varepsilon)\varepsilon \tag{5}$$

Proof. Let $f{:}V \to \mathsf{Pred}$ be a simple function. We use induction on the size of $\mathcal{E}$.

(1) Suppose $\mathcal{E} = \epsilon$. In that case the conclusion of the theorem follows immediately from assumption (2a).
(2) Let $\mathcal{E}$ be of the form $(\sigma X(d_X{:}D_X) = \phi)\ \mathcal{E}'$ for some $X \notin \mathsf{bnd}(\mathcal{E}')$. We assume as our induction hypothesis that for all environments η_1' and η_2', if the following conditions are met:
   (a) $\mathsf{bnd}(\mathcal{E}') \cup \mathsf{occ}(\mathcal{E}') \subseteq V$ and
   (b) for all $Y \in V$:
      (i) $[\![f(Y) \wedge Y(d_Y)]\!]\,\eta_1'\varepsilon = [\![f(Y) \wedge Y(d_Y)]\!]\,\eta_2'\varepsilon$
      (ii) $f(Y) \wedge \phi \leftrightarrow (f(Y) \wedge \phi)\left[_{X_i \in V}(f(X_i) \wedge X_i(d_{X_i}))_{\langle d_{X_i}\rangle}/X_i\right]$

   then for all $Y \in V$, we have

$$[\![f(Y) \wedge Y(d_Y)]\!]\,([\![\mathcal{E}']\!]\,\eta_1'\varepsilon)\varepsilon = [\![f(Y) \wedge Y(d_Y)]\!]\,([\![\mathsf{Apply}\,(f, \mathcal{E}')]\!]\,\eta_2'\varepsilon)\varepsilon$$

   Assume that the following holds:
   (a) $\mathsf{bnd}(\mathcal{E}) \cup \mathsf{occ}(\mathcal{E}) \subseteq V$ and
   (b) for all $Y \in V$:
      (i) $[\![f(Y) \wedge Y(d_Y)]\!]\,\eta_1\varepsilon = [\![f(Y) \wedge Y(d_Y)]\!]\,\eta_2\varepsilon$
      (ii) $f(Y) \wedge \phi \leftrightarrow (f(Y) \wedge \phi)\left[_{X_i \in V}(f(X_i) \wedge X_i(d_{X_i}))_{\langle d_{X_i}\rangle}/X_i\right]$

   We must show the below equivalence for all $Z \in V$:

$$[\![f(Z) \wedge Z(d_Z)]\!]\,([\![\mathcal{E}]\!]\,\eta_1\varepsilon)\varepsilon = [\![f(Z) \wedge Z(d_Z)]\!]\,([\![\mathsf{Apply}\,(f, \mathcal{E})]\!]\,\eta_2\varepsilon)\varepsilon \tag{6}$$

   Let $Z \in V$ be an arbitrary predicate variable. We continue as follows:

$$\begin{aligned}
&[\![f(Z) \wedge Z(d_Z)]\!]\,([\![\mathcal{E}]\!]\,\eta_1\varepsilon)\varepsilon\\
=\ &\{\text{Definition of } [\![\mathcal{E}]\!]\,\eta_1\varepsilon\}\\
&[\![f(Z) \wedge Z(d_Z)]\!]\,([\![\mathcal{E}']\!]\,\eta_1[\sigma\,\mathcal{X} \in [D_X \to B].\ [\![\phi_{\langle d_X\rangle}]\!]\,([\![\mathcal{E}']\!]\,\eta_1[\mathcal{X}/X]\varepsilon)/X]\varepsilon)\varepsilon
\end{aligned}$$

   Likewise, we derive:

$$\begin{aligned}
&[\![f(Z) \wedge Z(d_Z)]\!]\,([\![\mathsf{Apply}\,(f, \mathcal{E})]\!]\,\eta_2\varepsilon)\varepsilon\\
=\ &\{\text{Definition of } [\![\mathsf{Apply}\,(f, \mathcal{E})]\!]\,\eta_2\varepsilon\}\\
&[\![f(Z) \wedge Z(d_Z)]\!]\,\big([\![\mathsf{Apply}\,(f, \mathcal{E}')]\!]\\
&\quad \eta_2[\sigma\,\mathcal{X} \in [D_X \to B].\ [\![(f(X) \wedge \phi)_{\langle d_X\rangle}]\!]\,([\![\mathsf{Apply}\,(f, \mathcal{E}')]\!]\,\eta_2[\mathcal{X}/X]\varepsilon)/X]\varepsilon\big)\varepsilon
\end{aligned}$$

   From our assumption that $\mathsf{bnd}(\mathcal{E}) \cup \mathsf{occ}(\mathcal{E}) \subseteq V$, we immediately obtain $\mathsf{bnd}(\mathcal{E}') \cup \mathsf{occ}(\mathcal{E}') \subseteq V$, so for all $Z \neq X$, equation (6) follows from our induction hypothesis and assuming that it holds for $Z = X$. For the latter, i.e. for $Z = X$, we must demonstrate that:

$$\begin{aligned}
&[\![f(X) \wedge X(d_X)]\!]\,([\![\mathcal{E}']\!]\,\eta_1[\sigma\,\mathcal{X} \in [D_X \to B].\ [\![\phi_{\langle d_X\rangle}]\!]\,([\![\mathcal{E}']\!]\,\eta_1[\mathcal{X}/X]\varepsilon)/X]\varepsilon)\varepsilon\\
=\\
&[\![f(X) \wedge X(d_X)]\!]\,\big([\![\mathsf{Apply}\,(f, \mathcal{E}')]\!]\\
&\quad \eta_2[\sigma\,\mathcal{X} \in [D_X \to B].\ [\![(f(X) \wedge \phi)_{\langle d_X\rangle}]\!]\,([\![\mathsf{Apply}\,(f, \mathcal{E}')]\!]\,\eta_2[\mathcal{X}/X]\varepsilon)/X]\varepsilon\big)\varepsilon
\end{aligned}$$

An application of the definition of semantics for predicate formulae, taking into account that f is a simple function, yields the equivalent equivalence:

$$\begin{aligned}
&[\![f(X)]\!]\,\varepsilon \wedge (\sigma\,\mathcal{X} \in [D_X \to B].\ [\![\phi_{\langle d_X\rangle}]\!]\,([\![\mathcal{E}']\!]\,\eta_1[\mathcal{X}/X]\varepsilon))([\![d_X]\!]\,\varepsilon)\\
=\\
&[\![f(X)]\!]\,\varepsilon \wedge (\sigma\,\mathcal{X} \in [D \to B].\\
&\quad [\![(f(X) \wedge \phi)_{\langle d_X\rangle}]\!]\,([\![\mathsf{Apply}\,(f, \mathcal{E}')]\!]\,\eta_2[\mathcal{X}/X]\varepsilon))([\![d_X]\!]\,\varepsilon)
\end{aligned} \tag{7}$$

Using Lemma 32 and our assumptions, we find:

$$\begin{aligned}
&[\![f(X)]\!]\,\varepsilon \wedge (\sigma\,\mathcal{X} \in [D_X \to B].\ [\![\phi_{\langle d_X\rangle}]\!]\,([\![\mathcal{E}']\!]\,\eta_1[\mathcal{X}/X]\varepsilon))([\![d_X]\!]\,\varepsilon)\\
=\\
&[\![f(X)]\!]\,\varepsilon \wedge (\sigma\,\mathcal{X} \in [D \to B].\ [\![(f(X) \wedge \phi)_{\langle d_X\rangle}]\!]\,([\![\mathcal{E}']\!]\,\eta_1[\mathcal{X}/X]\varepsilon))([\![d_X]\!]\,\varepsilon)
\end{aligned}$$

Using Lemma 33, our assumptions and the induction hypothesis, we find:

$$\begin{aligned}
&[\![f(X)]\!]\,\varepsilon \wedge (\sigma\,\mathcal{X} \in [D_X \to B].\ [\![(f(X) \wedge \phi)_{\langle d_X\rangle}]\!]\,([\![\mathcal{E}']\!]\,\eta_1[\mathcal{X}/X]\varepsilon))([\![d_X]\!]\,\varepsilon)\\
=\\
&[\![f(X)]\!]\,\varepsilon \wedge (\sigma\,\mathcal{X} \in [D \to B].\\
&\quad [\![(f(X) \wedge \phi)_{\langle d_X\rangle}]\!]\,([\![\mathsf{Apply}\,(f, \mathcal{E}')]\!]\,\eta_2[\mathcal{X}/X]\varepsilon))([\![d_X]\!]\,\varepsilon)
\end{aligned}$$

By transitivity of equivalence, we find that equivalence (7) holds. □

As a corollary of this theorem, we find the following result:

Corollary 36. *Let $\mathcal{E}$ be an equation system and let $f:V \to \mathsf{Pred}$ be a global invariant for $\mathcal{E}$. Then for all predicate formulae ϕ with $\mathsf{occ}(\phi) \subseteq V$ and all environments η, ε, we have:*

$$\phi \leftrightarrow \phi\left[_{X_i \in V}(f(X_i) \wedge X_i(d_{X_i}))_{\langle d_{X_i}\rangle}/X_i\right]$$

implies

$$[\![\phi]\!]\,([\![\mathcal{E}]\!]\,\eta\varepsilon)\varepsilon = [\![\phi]\!]\,([\![\mathsf{Apply}\,(f,\mathcal{E})]\!]\,\eta\varepsilon)\varepsilon \quad \square$$

This means that for an equation system $\mathcal{E}$ and a global invariant f of $\mathcal{E}$, it does not matter whether we use $\mathcal{E}$ or its invariant-strengthened version $\mathsf{Apply}\,(f,\mathcal{E})$ to evaluate a predicate formula ϕ that is invariant under f. The corollary below is a variation on this scheme, which simplifies specific equations in an equation system by removing simple predicate formulae that turn out to be invariants:

Corollary 37. *Let $\mathcal{E} :\equiv \mathcal{E}_0\ (\sigma X(d_X{:}D_X) = f(X) \wedge \psi)\ \mathcal{E}_1$ be an equation system and let $f:V \to \mathsf{Pred}$ be a global invariant for $\mathcal{E}$. Then for all $Z \in \mathsf{bnd}(\mathcal{E})$ and all terms $e{:}D_Z$ for which $f(Z)[e/d_Z]$ holds, we have:*

$$\begin{aligned}&[\![\mathcal{E}_0\ (\sigma X(d_X{:}D_X) = f(X) \wedge \psi)\ \mathcal{E}_1]\!]\,\eta\varepsilon(Z)([\![e]\!]\,\varepsilon)\\ =\ &[\![\mathsf{Apply}\,(f,\mathcal{E}_0)\ (\sigma X(d_X{:}D_X) = \psi)\ \mathsf{Apply}\,(f,\mathcal{E}_1)]\!]\,\eta\varepsilon(Z)([\![e]\!]\,\varepsilon) \quad \square\end{aligned}$$

Corollary 37 is particularly useful when evaluating equations of the form

$$(\nu X(d{:}D) = f(X) \wedge \bigwedge_{i\in I} \forall e_i{:}E_i.\ \psi_i \implies X(g_i(d,e_i)))$$

This is illustrated by the following proposition:

Proposition 38. *Let $\mathcal{E}$ be an equation system. Let f be a global invariant for $\mathcal{E}$ and assume $\mathcal{E}$ contains an equation for X of the form:*

$$\left(\nu X(d{:}D) = f(X) \wedge \bigwedge_{i\in I} \mathsf{Q}_1\, e_i^1{:}E_i^1 \ldots \mathsf{Q}_{m_i}\, e_i^{m_i}{:}E_i^{m_i}.\ \psi_i \implies X(g_i(d, e_i^1, \ldots, e_i^{m_i}))\right) \tag{8}$$

where $\mathsf{Q}_j \in \{\forall, \exists\}$ for any j, and for all i, ψ_i are simple predicate formulae and g_i is a data term that depends only on the values of d and $e_i^1, \ldots, e_i^{m_i}$. Then X has the solution $f(X)$.

Proof. Note that the solution to equation (8) is at most $f(X)$. Furthermore, using Corollary 37, it suffices to solve the following equation instead:

$$\left(\nu X(d{:}D) = \bigwedge_{i\in I} \mathsf{Q}_1\, e_i^1{:}E_i^1 \ldots \mathsf{Q}_{m_i}\, e_i^{m_i}{:}E_i^{m_i}.\ \psi_i \implies X(g_i(d, e_i^1, \ldots, e_i^{m_i}))\right) \tag{9}$$

Note that this equation is closed, and, hence does not rely on the solution to other predicate variables. Using e.g. a symbolic approximation, Eq. (9) can be shown to have $\top$ as its solution. Since the solution to Eq. (8) and Eq. (9) coincide whenever $f(X)$ holds, it immediately follows that $f(X)$ is also the greatest solution to Eq. (8). $\square$

In the terminology of [15], Eq. (8) is a pattern that has solution $f(X)$. Note that this pattern is an instance of a generalisation of the unsolved pattern of [15]. This pattern turns out to be quite useful in the examples of Section 7.

5. Robustness

In Section 3, we illustrated that local invariants are not robust with respect to common PBES transformations. For instance, Example 23 illustrated that substitution causes identified local invariants of the original equation system to break. As we will prove next, the notion of global invariants is robust with respect to the operations migration, unfolding and substitution, listed in Section 2.3. More specifically, we show that the set of all possible invariants for a fixed equation system is unaffected by migration and it grows when unfoldings or substitutions are applied to the equation system. The latter is important, since this means that both manipulations aid in finding useful invariants.

Theorem 39. *Let $\mathcal{E} :\equiv \mathcal{E}_0\ (\sigma X(d_X{:}D_X) = \phi)\ \mathcal{E}_1\ \mathcal{E}_2$ be an equation system. Let $f:V \to \mathsf{Pred}$ be a global invariant for $\mathcal{E}$. Then f is also a global invariant for the equation system $\mathcal{F} :\equiv \mathcal{E}_0\ \mathcal{E}_1\ (\sigma X(d_X{:}D_X) = \phi)\ \mathcal{E}_2$.*

Proof. The conditions for f being a global invariant are independent of the order of the equations, and, hence, any permutation of the equations preserves the global invariant. $\square$

An interesting observation that follows from Theorem 39 is the fact that invariants and solutions to equation systems are two independent properties. While invariants characterise the dependence of predicate variable instantiations on other predicate variable instantiations, it does not dictate solutions to these predicate variable instantiations. In fact, the notion of an invariant is insensitive to the chosen fixed points for the equations. On the other hand, the order of the equations and the fixed point signs are main concepts for determining the solution to an equation system. Below we give an example that shows that systems with the same set of invariants do not necessarily share solutions.

Example 40. Consider the following two equation systems ($\mu X(n{:}N) = X(n{+}1)$) and ($\nu X(n{:}N) = X(n{+}1)$). Both equations have exactly the same set of invariants, since the invariant conditions of Definition 24 are identical. Their solutions, however, are quite different: $X(n) = \bot$ for the first system and $X(n) = \top$ for the second one. □

Contrary to the operation of migration, unfolding and substitution modify the right-hand sides of an equation: both unfolding and substitution involve replacing predicate variables with the right-hand side expressions of the corresponding equation. The difference between unfolding and substitution is that unfolding operates locally and substitution is a global operation. The following lemma proves the stability of invariants under replacing variables with their corresponding right-hand side expressions.

Lemma 41. *Let $\mathcal{E}$ be an equation system and let $f{:}V \to \mathsf{Pred}$ be a global invariant for $\mathcal{E}$. For any predicate variable $X \in \mathsf{bnd}(\mathcal{E})$, we denote the right-hand side of X's defining equation in $\mathcal{E}$ by ϕ_X. Then, for all predicate variables $X, Y \in \mathsf{bnd}(\mathcal{E})$:*

$$\begin{array}{ll} & f(X) \wedge \phi_X[\phi_{Y\langle d_Y\rangle}/Y] \\ \leftrightarrow & (f(X) \wedge \phi_X[\phi_{Y\langle d_Y\rangle}/Y])\ \left[_{Z\in V}(f(Z) \wedge Z(d_Z))_{\langle d_Z\rangle}/Z\right] \end{array}$$

Proof. We calculate, using properties proved previously, starting from the right-hand side of the desired equality:

$$\begin{array}{ll} & (f(X) \wedge \phi_X[\phi_{Y\langle d_Y\rangle}/Y])\ \left[_{Z\in V}(f(Z) \wedge Z(d_Z))_{\langle d_Z\rangle}/Z\right] \\ \leftrightarrow & \{V = (V \setminus \{Y\}) \cup \{Y\}\} \\ & \Big((f(X) \wedge \phi_X[\phi_{Y\langle d_Y\rangle}/Y]) \\ & \quad \left[_{Z\in V\setminus\{Y\}}(f(Z) \wedge Z(d_Z))_{\langle d_Z\rangle}/Z\right]\Big)\ [(f(Y) \wedge Y(d_Y))_{\langle d_Y\rangle}/Y] \\ \leftrightarrow & \{\text{Distributivity of substitution over } \wedge, f \text{ is simple;}\} \\ & \{\text{Lemma 13 successively applied to all } Z \in V \setminus \{Y\}\text{; Lemma 10}\} \\ & \Big(f(X) \wedge \phi_X\ \left[_{Z\in V\setminus\{Y\}}(f(Z) \wedge Z(d_Z))_{\langle d_Z\rangle}/Z\right] \\ & \quad [\phi_{Y\langle d_Y\rangle}\ \left[_{Z\in V\setminus\{Y\}}(f(Z) \wedge Z(d_Z))_{\langle d_Z\rangle}/Z\right]/Y]\Big)\ [(f(Y) \wedge Y(d_Y))_{\langle d_Y\rangle}/Y] \\ \leftrightarrow & \{\text{Distributivity of substitution over } \wedge, f \text{ is simple;}\} \\ & \{\text{Lemma 12, } (V \setminus \{Y\}) \cup \{Y\} = V\} \\ & \Big(f(X) \wedge \phi_X\ \left[_{Z\in V\setminus\{Y\}}(f(Z) \wedge Z(d_Z))_{\langle d_Z\rangle}/Z\right]\Big) \\ & \quad \left[\phi_{Y\langle d_Y\rangle}\ \left[_{Z\in V}(f(Z) \wedge Z(d_Z))_{\langle d_Z\rangle}/Z\right]/Y\right] \\ \leftrightarrow & \{\text{Proposition 25, Lemma 10}\} \\ & (f(X) \wedge \phi_X\ [(f(Y) \wedge Y(d_Y))_{\langle d_Y\rangle}/Y]) \\ & \quad [\phi_{Y\langle d_Y\rangle}\ \left[_{Z\in V}(f(Z) \wedge Z(d_Z))_{\langle d_Z\rangle}/Z\right]/Y] \\ \leftrightarrow & \{\text{Distributivity, } f \text{ is simple, Lemma 12}\} \\ & (f(X) \wedge \phi_X)\ \left[(f(Y) \wedge \phi_{Y\langle d_Y\rangle}\ \left[_{Z\in V}(f(Z) \wedge Z(d_Z))_{\langle d_Z\rangle}/Z\right])_{\langle d_Y\rangle}/Y\right] \\ \leftrightarrow & \{\text{Proposition 25, Lemma 10}\} \\ & (f(X) \wedge \phi_X)[(f(Y) \wedge \phi_{Y\langle d_Y\rangle})/Y] \\ \leftrightarrow & \{\text{Lemma 12}\} \\ & (f(X) \wedge \phi_X)[(f(Y) \wedge Y(d_Y))_{\langle d_Y\rangle}/Y][\phi_{Y\langle d_Y\rangle}/Y] \\ \leftrightarrow & \{\text{Proposition 25: } (f(X) \wedge \phi_X)[(f(Y) \wedge Y(d_Y))_{\langle d_Y\rangle}/Y] = f(X) \wedge \phi_X\} \\ & \{\text{Distributivity, } f \text{ is simple}\} \\ & f(X) \wedge \phi_X[\phi_{Y\langle d_Y\rangle}/Y] \quad \square \end{array}$$

The robustness of global invariants with respect to substitution and unfolding follows from here.

Theorem 42. *Let $\mathcal{E} :\equiv \mathcal{E}_0\ (\sigma X(d_X{:}D_X) = \phi)\ \mathcal{E}_1$ be an equation system and let $f{:}V \to \mathsf{Pred}$ a global invariant for $\mathcal{E}$. Then f is also a global invariant for the equation system $\mathcal{F} :\equiv \mathcal{E}_0\ (\sigma X(d_X{:}D_X) = \phi[\phi_{\langle d_X\rangle}/X])\ \mathcal{E}_1$.*

Proof. The invariant conditions for predicate variables $Y \neq X$ are immediately satisfied by f for $\mathcal{F}$, since they coincide with those for f and $\mathcal{E}$. For X, the invariant condition is

$$f(X) \wedge \phi[\phi_{\langle d_X\rangle}/X] \leftrightarrow\ (f(X) \wedge \phi[\phi_{\langle d_X\rangle}/X])\ \left[_{Z\in V}(f(Z) \wedge Z(d_Z))_{\langle d_Z\rangle}/Z\right],$$

which follows immediately from Lemma 41 by taking $Y = X$. □

The reverse of Theorem 42 does not hold, which means that unfolding equations in an equation system increases the set of global invariants that holds for the original equation system. Below is an example to illustrate this fact:

Example 43. Let $\nu X(n{:}N) = X(n+1)$ be an equation system. Using unfolding, we obtain the following equivalent equation system: $\nu X(n{:}N) = X(n+2)$. Clearly, the function f that assigns to X the predicate formula $\mathsf{even}(n)$ is a global invariant for the latter equation. However, f is not a global invariant for the original equation. Therefore, by unfolding the set of invariants for an equation system increases. □

Theorem 44. *Let* $\mathcal{E} :\equiv \mathcal{E}_0\ (\sigma X(d_X{:}D_X) = \phi)\ \mathcal{E}_1\ (\sigma' Y(d_Y{:}D_Y) = \psi)\ \mathcal{E}_2$ *and* $\mathcal{F} :\equiv \mathcal{E}_0\ (\sigma X(d_X{:}D_X) = \phi[\psi_{\langle d_Y\rangle}/Y])\ \mathcal{E}_1\ (\sigma' Y(d_Y{:}D_Y) = \psi)\ \mathcal{E}_2$ *be equation systems. If* $f{:}V \to \mathsf{Pred}$ *is a global invariant for* $\mathcal{E}$ *then* f *is also a global invariant for* $\mathcal{F}$.

Proof. The conditions for f to be an invariant in $\mathcal{F}$ do not change for variables $Z \neq X$. We only have to prove that

$$f(X) \wedge \phi[\psi_{\langle d_Y\rangle}/Y] \leftrightarrow (f(X) \wedge \phi[\psi_{\langle d_Y\rangle}/Y])\ \left[_{Z\in V}(f(Z) \wedge Z(d_Z))_{\langle d_Z\rangle}/Z\right].$$

This follows immediately from Lemma 41. □

Observe that one can equally well show that substituting in the other direction (i.e. substituting ϕ for X in the equation of Y in Theorem 44) does not violate the invariant conditions. However, such an operation in general affects the solution of the equation system and is therefore not a sound manipulation on equation systems. Note that substitution also strictly adds invariants, as illustrated by the following example.

Example 45. Consider the system $(\mu X(n{:}N) = Y(n+1))\ (\mu Y(n{:}N) = X(2n))$. The simple function $f(X) = f(Y) = \mathsf{even}(n)$ is not a global invariant of this system. After a backward substitution, we obtain the equivalent system $(\mu X(n{:}N) = X(2(n+1)))\ (\mu Y(n{:}N) = X(2n))$, for which f is a global invariant. □

6. Process invariants

Invariants traditionally have been used in program and process verification to reason about (the correctness of) recursive and iterative programs and processes, and, in particular, about safety requirements. In this section, we claim a precise correspondence between the notion of invariants for processes (cf. [2]) and invariants for equation systems.

6.1. Specification languages

Linear process equations (LPEs) have been proposed as *symbolic* representations of general (infinite) labelled transition systems, the semantic framework for specifying and analysing complex, reactive systems. In an LPE, the state of a process is modelled by a finite vector of (possibly infinite) sorted variables, and the behaviour is described by a finite set R of condition–action–effect rules, among which one selects one non-deterministically (denoted by $\sum R$); each condition–action–effect rule $r \in R$ can depend on non-deterministically chosen values for "local" variables (denoted by the quantifier $\sum_{e:E} r$). Note that the apparent restrictiveness of the format of the LPE does not incur a loss of expressive power in general. Many process languages that include more complex process operators, such as parallelism, enjoy the nice property that all relevant processes described in that language can be transformed into LPEs (although sometimes at the cost of extra complexity in the data structures). Prime examples of such languages are μCRL [13] and mCRL2 [12].

Definition 46. A *linear process equation* is a parameterised equation taking the form

$$P(d{:}D) = \sum \left\{ \sum_{e_{\mathsf{a}}:E_{\mathsf{a}}} c_{\mathsf{a}}(d, e_{\mathsf{a}}) \Longrightarrow \mathsf{a}(f_{\mathsf{a}}(d, e_{\mathsf{a}})) \cdot P(g_{\mathsf{a}}(d, e_{\mathsf{a}})) \mid \mathsf{a} \in \mathcal{A}ct \right\}$$

where $f_{\mathsf{a}}{:}D \times E_{\mathsf{a}} \to D_{\mathsf{a}}$, $g_{\mathsf{a}}{:}D \times E_{\mathsf{a}} \to D$ and $c_{\mathsf{a}}{:}D \times E_{\mathsf{a}} \to B$ for each action label $\mathsf{a} \in \mathcal{A}ct$. Note that here D, D_{a} and E_{a} are general data sorts. The restrictions to single sorts D and E_{a} is again done for brevity and does not cause a loss of generality.

In the above definition, the LPE P specifies, for each action name $\mathsf{a} \in \mathcal{A}ct$, that if in the current state d the condition $c_{\mathsf{a}}(d, e_{\mathsf{a}})$ holds, for an arbitrary e_{a} of sort E_{a}, then action $\mathsf{a}(f_{\mathsf{a}}(d, e_{\mathsf{a}}))$ is enabled and the effect of executing this action is that the state is changed to $g_{\mathsf{a}}(d, e_{\mathsf{a}})$. Thus, the values of the condition, action parameter and new state may depend on the current state and a chosen value for variable e_{a}. This intuition is formalised by the semantics of LPEs, defined in terms of *labelled transition systems*. Hereafter, we assume a fixed, arbitrary LPE P, given by Definition 46.

Definition 47. The *labelled transition system* of the LPE P of Definition 46 with initial state d_0 is a quadruple $\mathcal{M} = \langle S, \Sigma, \to, s_0\rangle$, where

- $S = \{v \mid v \in D\}$ is the set of states; $s_0 = d_0$ is the initial state,
- $\Sigma = \{\mathsf{a}(v) \mid \mathsf{a} \in \mathcal{A}ct \wedge v \in D_{\mathsf{a}}\}$ is the (possibly infinite) set of actions,
- $\to = \{(d, \mathsf{a}(v), d') \mid \mathsf{a} \in \mathcal{A}ct \wedge \exists e_{\mathsf{a}} \in E_{\mathsf{a}}.\ c_{\mathsf{a}}(d, e_{\mathsf{a}}) \wedge v = f_{\mathsf{a}}(d, e_{\mathsf{a}}) \wedge d' = g_{\mathsf{a}}(d, e_{\mathsf{a}})\}$ is the transition relation.

An invariant of P is a simple formula ι that is closed under the next-step relation of the LPE: provided that ι holds for a state d, it also holds for all states $g_{\mathtt{a}}(d, e_{\mathtt{a}})$ that are reachable from d via enabled actions $\mathtt{a}$. Invariants are useful for quickly verifying certain safety properties.

Definition 48. A simple predicate ι is an *invariant* of P iff the following ordering holds for all actions $\mathtt{a} \in \mathcal{A}ct$:

$$\iota \wedge c_{\mathtt{a}}(d, e_{\mathtt{a}}) \rightarrow (\iota[g_{\mathtt{a}}(d, e_{\mathtt{a}})/d])$$

where $c_{\mathtt{a}}(d, e_{\mathtt{a}})$ and $g_{\mathtt{a}}(d, e_{\mathtt{a}})$ are taken syntactically from P.

Example 49. To illustrate the notion of a process invariant, consider the following LPE:

$$\begin{aligned} P(n{:}N) \;&= \sum_{m:N} m \geq n \Longrightarrow \mathtt{r}(m) \cdot P(m) \\ &\quad + \top \Longrightarrow \mathtt{s}(n) \cdot P(n) \end{aligned}$$

LPE P reads an integer into its buffer that is at least as large as its current integer, and, is at any moment able to output the value currently in the buffer. An obvious invariant for P is the simple predicate formula $n \geq 10$, since both $n \geq 10 \wedge m \geq n \rightarrow m \geq 10$ and $n \geq 10 \wedge \top \rightarrow n \geq 10$ hold. □

6.2. First-order modal μ-calculus

In [20,11], a modal language for verification of data-dependent process languages is defined. The language is called the *first-order modal μ-calculus*, hereafter referred to as the μ-calculus. As suggested by the name, the language is a first-order extension of the standard modal μ-calculus due to Kozen [18]. The extension permits the use of data variables and parameters to capture the essential data dependencies in the process behaviour. The grammar of the calculus is given by the following rules:

$$\begin{aligned} \phi &::= b \mid X(e) \mid \phi \oplus \phi \mid \mathtt{Q}\, d{:}D.\ \phi \mid [\alpha]\phi \mid \langle\alpha\rangle\phi \mid (\sigma X(d_f{:}D_f := e).\ \phi) \\ \alpha &::= b \mid \mathtt{a}(e) \mid \neg\alpha \mid \alpha \wedge \alpha \mid \forall d{:}D.\alpha \end{aligned}$$

where σ is a least or greatest fixed point sign, and $\oplus \in \{\wedge, \vee\}$ and $\mathtt{Q} \in \{\forall, \exists\}$ are used as abbreviations from hereon. The semantics of μ-calculus formulae is defined over an LTS, induced by an LPE P and requires environments assigning values to fixed point variables X and data variables d. We only consider fixed point formulae in normal form, i.e. formulae for which every fixed point variable is bound at most once and every occurrence of a fixed point variable is bound.

We assume an interpretation function $[\![\_]\!]_P^{\theta\varepsilon}$ for μ-calculus formulae, in which P is an LPE and θ and ε are fixed point variable environments and data variable environments, respectively. The interpretation maps a formula ϕ onto a set of states of the LTS induced by P. For a formal definition of the semantics, we refer to [20,11,14].

The global model checking problem $P \models \Phi$ and the local model checking problem $P(e) \models \Phi$, where e is an initial value for the P and Φ is a μ-calculus formula, can be translated to the problem of solving an equation system [20,11,14]. The transformation is given in Table 1 and is described in detail in [14]. It assumes that Φ is of the form $\sigma X(d_f{:}D_f := e).\ \psi$, where the fixed point X is possibly effectless.

Lemma 50. *Let $\iota \in \mathsf{Pred}$ be an invariant for the LPE P. Let Φ be an arbitrary μ-calculus formula and ψ an arbitrary subformula of Φ. Let V be the set of fixed point variables that are bound by a fixed point in Φ. Then:*

$$\iota \wedge \mathbf{RHS}_\Phi(\psi) \leftrightarrow (\iota \wedge \mathbf{RHS}_\Phi(\psi))\ \left[_{\tilde{Z}\in V}(\iota \wedge \tilde{Z}(d_{\tilde{Z}}))_{\langle d_{\tilde{Z}}\rangle}/\tilde{Z}\right]$$

Proof. The proof is by induction on the structure of ψ. The base cases are addressed below:

- Case $\psi \equiv b$. Then:

$$\begin{aligned} &\iota \wedge \mathbf{RHS}_\Phi(b) \\ \leftrightarrow\ &\{\text{Definition}\} \\ &\iota \wedge b \\ \leftrightarrow\ &\{\text{Syntactic substitution is effectless on simple predicate formulae}\} \\ &(\iota \wedge b)\ \left[_{\tilde{Z}\in V}(\iota \wedge \tilde{Z}(d_{\tilde{Z}}))_{\langle d_{\tilde{Z}}\rangle}/\tilde{Z}\right] \\ \leftrightarrow\ &\{\text{Definition}\} \\ &(\iota \wedge \mathbf{RHS}_\Phi(b))\ \left[_{\tilde{Z}\in V}(\iota \wedge \tilde{Z}(d_{\tilde{Z}}))_{\langle d_{\tilde{Z}}\rangle}/\tilde{Z}\right] \end{aligned}$$

Table 1
Inductive translation scheme for encoding the problem $P \models \Phi$, where $\Phi = \sigma X(d_f{:}D_f := e).\ \psi$, into the closed equation system $\mathbf{E}(\Phi)$.

$$
\begin{aligned}
\mathbf{E}(b) &= \epsilon\\
\mathbf{E}(X(e)) &= \epsilon\\
\mathbf{E}(\phi_1 \oplus \phi_2) &= \mathbf{E}(\phi_1)\ \mathbf{E}(\phi_2)\\
\mathbf{E}(\mathsf{Q}\, d{:}D.\phi) &= \mathbf{E}(\phi)\\
\mathbf{E}([\alpha]\phi) &= \mathbf{E}(\phi)\\
\mathbf{E}(\langle\alpha\rangle\phi) &= \mathbf{E}(\phi)\\
\mathbf{E}(\sigma X(d_f{:}D_f := e).\ \psi) &= (\sigma\tilde{X}(d{:}D, d_f{:}D_f, \mathbf{Par}_{[]}(X,\Phi)) = \mathbf{RHS}_{\Phi}(\psi))\\
&\quad\ \mathbf{E}(\phi)
\end{aligned}
$$

$$
\begin{aligned}
\mathbf{RHS}_{\Phi}(b) &= b\\
\mathbf{RHS}_{\Phi}(X(e)) &= \tilde{X}(d, e, \mathbf{Par}_{[]}(X,\Phi))\\
\mathbf{RHS}_{\Phi}(\phi_1 \oplus \phi_2) &= \mathbf{RHS}_{\Phi}(\phi_1) \oplus \mathbf{RHS}_{\Phi}(\phi_2)\\
\mathbf{RHS}_{\Phi}(\mathsf{Q}\, d{:}D.\phi) &= \mathsf{Q}\, d{:}D.\ \mathbf{RHS}_{\Phi}(\psi)\\
\mathbf{RHS}_{\Phi}([\alpha]\phi) &= \textstyle\bigwedge_{\mathsf{a}\in\mathcal{A}ct}\ \forall e_{\mathsf{a}}{:}D_{\mathsf{a}}.\\
&\quad (c_{\mathsf{a}}(d, e_{\mathsf{a}}) \wedge \mathsf{match}(\mathsf{a}(f_{\mathsf{a}}(d, e_{\mathsf{a}})), \alpha))\\
&\quad\ \Longrightarrow\ (\mathbf{RHS}_{\Phi}(\phi)[g_{\mathsf{a}}(d, e_{\mathsf{a}})/d])\\
\mathbf{RHS}_{\Phi}(\langle\alpha\rangle\phi) &= \textstyle\bigvee_{\mathsf{a}\in\mathcal{A}ct}\ \exists e_{\mathsf{a}}{:}D_{\mathsf{a}}.\\
&\quad (c_{\mathsf{a}}(d, e_{\mathsf{a}}) \wedge \mathsf{match}(\mathsf{a}(f_{\mathsf{a}}(d, e_{\mathsf{a}})), \alpha)\\
&\quad\ \wedge\ (\mathbf{RHS}_{\Phi}(\phi)[g_{\mathsf{a}}(d, e_{\mathsf{a}})/d]))\\
\mathbf{RHS}_{\Phi}(\sigma X(d_f{:}D_f := e).\ \phi) &= \tilde{X}(d, e, \mathbf{Par}_{[]}(X,\Phi))
\end{aligned}
$$

$$
\begin{aligned}
\mathsf{match}(\mathsf{a}(v), b) &= b\\
\mathsf{match}(\mathsf{a}(v), \mathsf{a}(d)) &= v = d\\
\mathsf{match}(\mathsf{a}(v), \mathsf{a}'(d)) &= \bot\\
\mathsf{match}(\mathsf{a}(v), \neg\alpha) &= \neg\mathsf{match}(\mathsf{a}(v), \alpha)\\
\mathsf{match}(\mathsf{a}(v), \alpha_1 \wedge \alpha_2) &= \mathsf{match}(\mathsf{a}(v), \alpha_1) \wedge \mathsf{match}(\mathsf{a}(v), \alpha_2)\\
\mathsf{match}(\mathsf{a}(v), \forall d{:}D.\ \alpha) &= \forall d{:}D.\ \mathsf{match}(\mathsf{a}(v), \alpha)
\end{aligned}
$$

$$
\begin{aligned}
\mathbf{Par}_l(X, b) &= []\\
\mathbf{Par}_l(X, X(e)) &= []\\
\mathbf{Par}_l(X, \phi_1 \oplus \phi_2) &= \mathbf{Par}_l(X, \phi_1) +\!\!+ \mathbf{Par}_l(X, \phi_2)\\
\mathbf{Par}_l(X, \mathsf{Q}\, d{:}D.\ \phi) &= \mathbf{Par}_{[d:D]+\!\!+l}(X, \phi)\\
\mathbf{Par}_l(X, [\alpha]\phi) &= \mathbf{Par}_l(X, \phi)\\
\mathbf{Par}_l(X, \langle\alpha\rangle\phi) &= \mathbf{Par}_l(X, \phi)\\
\mathbf{Par}_l(X, \sigma Z(d_f{:}D_f := e).\ \phi) &= \begin{cases} l & \text{if } Z = X\\ \mathbf{Par}_{[d_f:D_f]+\!\!+l}(X, \phi) & \text{otherwise}\end{cases}
\end{aligned}
$$

- Case $\psi \equiv X(e)$. Then:

$$
\begin{aligned}
&\iota \wedge \mathbf{RHS}_{\Phi}(X(e))\\
\leftrightarrow\ &\{\text{Definition}\}\\
&\iota \wedge \tilde{X}(d, e, \mathbf{Par}_{[]}(X,\Phi))\\
\leftrightarrow\ &\{\text{Idempotence of } \wedge\}\\
&\iota \wedge (\iota \wedge \tilde{X}(d, e, \mathbf{Par}_{[]}(X,\Phi)))\\
\leftrightarrow\ &\{\text{Definition of syntactic substitution}\}\\
&\iota \wedge \left(\mathbf{RHS}_{\Phi}(X(e))\ \left[_{\tilde{Z}\in V}(\iota \wedge \tilde{Z}(d_{\tilde{Z}}))_{\langle d_{\tilde{Z}}\rangle}/\tilde{Z}\right]\right)\\
\leftrightarrow\ &\{\iota \text{ is a simple predicate}\}\\
&(\iota \wedge \mathbf{RHS}_{\Phi}(X(e)))\ \left[_{\tilde{Z}\in V}(\iota \wedge \tilde{Z}(d_{\tilde{Z}}))_{\langle d_{\tilde{Z}}\rangle}/\tilde{Z}\right]
\end{aligned}
$$

As our inductive hypothesis, we assume that for any formula Φ and the subformulae ψ_i we have:

$$\iota \wedge \mathbf{RHS}_{\Phi}(\psi_i) \leftrightarrow (\iota \wedge \mathbf{RHS}_{\Phi}(\psi_i))\ \left[_{\tilde{Z}\in V}(\iota \wedge \tilde{Z}(d_{\tilde{Z}}))_{\langle d_{\tilde{Z}}\rangle}/\tilde{Z}\right] \qquad \text{(IH)}$$

- Case $\psi \equiv \psi_1 \oplus \psi_2$. Then

$$
\begin{array}{ll}
 & \iota \wedge \mathbf{RHS}_\Phi(\psi_1 \oplus \psi_2) \\
\leftrightarrow & \{\text{Definition of } \mathbf{RHS}_\Phi(\psi_1 \oplus \psi_2), \alpha \wedge (\beta \oplus \gamma) = (\alpha \oplus \beta) \wedge (\alpha \oplus \gamma)\} \\
 & (\iota \wedge \mathbf{RHS}_\Phi(\psi_1)) \oplus (\iota \wedge \mathbf{RHS}_\Phi(\psi_2)) \\
\leftrightarrow & \{\text{Induction Hypothesis}\} \\
 & (\iota \wedge \mathbf{RHS}_\Phi(\psi_1)) \left[_{\tilde{Z} \in V} (\iota \wedge \tilde{Z}(d_{\tilde{Z}}))_{\langle d_{\tilde{Z}} \rangle} / \tilde{Z}\right] \\
 & \oplus\ (\iota \wedge \mathbf{RHS}_\Phi(\psi_2)) \left[_{\tilde{Z} \in V} (\iota \wedge \tilde{Z}(d_{\tilde{Z}}))_{\langle d_{\tilde{Z}} \rangle} / \tilde{Z}\right] \\
\leftrightarrow & \{\text{Definition of syntactic substitution}\} \\
 & ((\iota \wedge \mathbf{RHS}_\Phi(\psi_1)) \oplus (\iota \wedge \mathbf{RHS}_\Phi(\psi_2))) \left[_{\tilde{Z} \in V} (\iota \wedge \tilde{Z}(d_{\tilde{Z}}))_{\langle d_{\tilde{Z}} \rangle} / \tilde{Z}\right] \\
\leftrightarrow & \{\text{Definition of } \mathbf{RHS}_\Phi(\psi_1 \oplus \psi_2), \alpha \wedge (\beta \oplus \gamma) = (\alpha \oplus \beta) \wedge (\alpha \oplus \gamma)\} \\
 & (\iota \wedge \mathbf{RHS}_\Phi(\psi_1 \oplus \psi_2)) \left[_{\tilde{Z} \in V} (\iota \wedge \tilde{Z}(d_{\tilde{Z}}))_{\langle d_{\tilde{Z}} \rangle} / \tilde{Z}\right]
\end{array}
$$

- Case $\phi \equiv \mathsf{Q}\, d{:}D.\ \psi_1$. Then

$$
\begin{array}{ll}
 & \iota \wedge \mathbf{RHS}_\Phi(\mathsf{Q}\, d{:}D.\ \psi_1) \\
\leftrightarrow & \{\text{Definition of } \mathbf{RHS}_\Phi(\mathsf{Q}\, d{:}D.\ \psi_1)\} \\
 & \iota \wedge \mathsf{Q}\, d{:}D.\ \mathbf{RHS}_\Phi(\psi_1) \\
\leftrightarrow & \{\text{Variable } d \text{ does not occur in } \iota\} \\
 & \mathsf{Q}\, d{:}D.\ \iota \wedge \mathbf{RHS}_\Phi(\psi_1) \\
\leftrightarrow & \{\text{Induction Hypothesis}\} \\
 & \mathsf{Q}\, d{:}D.\ (\iota \wedge \mathbf{RHS}_\Phi(\psi_1)) \left[_{\tilde{Z} \in V} (\iota \wedge \tilde{Z}(d_{\tilde{Z}}))_{\langle d_{\tilde{Z}} \rangle} / \tilde{Z}\right] \\
\leftrightarrow & \{\text{Syntactic substitution is effectless for } \iota\text{; variable } d \text{ does not occur in } \iota\} \\
 & (\iota \wedge \mathsf{Q}\, d{:}D.\ \mathbf{RHS}_\Phi(\psi_1)) \left[_{\tilde{Z} \in V} (\iota \wedge \tilde{Z}(d_{\tilde{Z}}))_{\langle d_{\tilde{Z}} \rangle} / \tilde{Z}\right] \\
\leftrightarrow & \{\text{Definition of } \mathbf{RHS}_\Phi(\mathsf{Q}\, d{:}D.\ \psi_1)\} \\
 & (\iota \wedge \mathbf{RHS}_\Phi(\mathsf{Q}\, d{:}D.\ \psi_1)) \left[_{\tilde{Z} \in V} (\iota \wedge \tilde{Z}(d_{\tilde{Z}}))_{\langle d_{\tilde{Z}} \rangle} / \tilde{Z}\right]
\end{array}
$$

- Case $\psi \equiv [\alpha]\psi_1$. Then

$$
\begin{array}{ll}
 & \iota \wedge \mathbf{RHS}_\Phi([\alpha]\psi_1) \\
\leftrightarrow & \iota \wedge \forall e_{\mathsf{a}}{:}E_{\mathsf{a}}.(c_{\mathsf{a}}(d, e_{\mathsf{a}}) \wedge \mathsf{match}(\mathsf{a}(f_{\mathsf{a}}(d, e_{\mathsf{a}})), \alpha)) \Rightarrow (\mathbf{RHS}_\Phi(\psi_1)[g_{\mathsf{a}}(d, e_{\mathsf{a}})/d]) \\
\leftrightarrow & \{\iota \text{ is a process invariant, thus } \iota \wedge c_{\mathsf{a}}(d, e_{\mathsf{a}}) \Rightarrow \iota[g_{\mathsf{a}}(d, e_{\mathsf{a}})/d]\} \\
 & \iota \wedge \forall e_{\mathsf{a}}{:}E_{\mathsf{a}}.(c_{\mathsf{a}}(d, e_{\mathsf{a}}) \wedge \mathsf{match}(\mathsf{a}(f_{\mathsf{a}}(d, e_{\mathsf{a}})), \alpha)) \\
 & \Rightarrow (\iota \wedge \mathbf{RHS}_\Phi(\psi_1))[g_{\mathsf{a}}(d, e_{\mathsf{a}})/d]) \\
\leftrightarrow & \{\text{Induction Hypothesis}\} \\
 & \iota \wedge \forall e_{\mathsf{a}}{:}E_{\mathsf{a}}.(c_{\mathsf{a}}(d, e_{\mathsf{a}}) \wedge \mathsf{match}(\mathsf{a}(f_{\mathsf{a}}(d, e_{\mathsf{a}})), \alpha)) \Rightarrow \\
 & \quad (\iota \wedge \mathbf{RHS}_\Phi(\psi_1)) \left[_{\tilde{Z} \in V} (\iota \wedge \tilde{Z}(d_{\tilde{Z}}))_{\langle d_{\tilde{Z}} \rangle} / \tilde{Z}\right] [g_{\mathsf{a}}(d, e_{\mathsf{a}})/d] \\
\leftrightarrow & \{\text{Rewriting}\} \\
 & (\iota \wedge \forall e_{\mathsf{a}}{:}E_{\mathsf{a}}.(c_{\mathsf{a}}(d, e_{\mathsf{a}}) \wedge \mathsf{match}(\mathsf{a}(f_{\mathsf{a}}(d, e_{\mathsf{a}})), \alpha)) \Rightarrow \\
 & \quad (\iota \wedge \mathbf{RHS}_\Phi(\psi_1))[g_{\mathsf{a}}(d, e_{\mathsf{a}})/d]) \left[_{\tilde{Z} \in V} (\iota \wedge \tilde{Z}(d_{\tilde{Z}}))_{\langle d_{\tilde{Z}} \rangle} / \tilde{Z}\right] \\
\leftrightarrow & \{\iota \text{ is a process invariant, thus } \iota[g_{\mathsf{a}}(d, e_{\mathsf{a}})/d] \text{ can be removed}\} \\
 & (\iota \wedge \forall e_{\mathsf{a}}{:}E_{\mathsf{a}}.(c_{\mathsf{a}}(d, e_{\mathsf{a}}) \wedge \mathsf{match}(\mathsf{a}(f_{\mathsf{a}}(d, e_{\mathsf{a}})), \alpha)) \Rightarrow \\
 & \quad (\mathbf{RHS}_\Phi(\psi_1)[g_{\mathsf{a}}(d, e_{\mathsf{a}})/d])) \left[_{\tilde{Z} \in V} (\iota \wedge \tilde{Z}(d_{\tilde{Z}}))_{\langle d_{\tilde{Z}} \rangle} / \tilde{Z}\right] \\
\leftrightarrow & \{\text{Definition of } \mathbf{RHS}_\Phi([\alpha]\psi_1)\} \\
 & (\iota \wedge \mathbf{RHS}_\Phi([\alpha]\psi_1)) \left[_{\tilde{Z} \in V} (\iota \wedge \tilde{Z}(d_{\tilde{Z}}))_{\langle d_{\tilde{Z}} \rangle} / \tilde{Z}\right]
\end{array}
$$

- Case $\psi \equiv \langle\alpha\rangle\psi_1$. Similar to the case $\psi \equiv [\alpha]\psi_1$.
- Case $\psi \equiv \sigma X(d_f{:}D_f := e).\psi_1$. Similar to the base case $\psi \equiv X(e)$. □

The above lemma proves that in the translation of a μ-calculus formula to a predicate formula, process invariants satisfy the global invariance conditions. Ultimately, this means that process invariants are preserved by the transformation of the model checking problem. This is stated in the below theorem.

Theorem 51. *Let Φ be a μ-calculus formula. Let ι be a process invariant for the LPE P. Then the simple function defined by $(\lambda X{\in}\mathsf{bnd}(\mathbf{E}(\Phi)).\ \iota)$ is a global invariant of $\mathbf{E}(\Phi)$.*

Proof. Follows from Lemma 50 and the translation $\mathbf{E}(\phi)$. □

Note that the reverse of Theorem 51 does not hold: if f is a global invariant for an equation system $\mathbf{E}(\Phi)$ for some formula Φ and LPE P, then f does not necessarily lead to an invariant for the process P (see the below example). This supports the viewpoint that the analysis on the level of equation systems is in general more powerful than the analysis on the level of processes.

Example 52. Consider the process that models the stock value of some company and reports its current value if asked. Initially, the stock value is some value larger than threshold T.

$$\begin{aligned} M(v{:}N) \;=\; & \sum_{m:N} \mathtt{up} \cdot M(v+m) \\ & + \sum_{m:N} m \leq v \implies \mathtt{down} \cdot M(v-m) \\ & + \mathtt{current}(v) \cdot M(v) \end{aligned}$$

We verify that when the stock value does not decrease, the value that is reported is always above threshold T: $\nu X.[\neg \mathtt{down}]X \wedge \forall n{:}N.[\mathtt{current}(n)](n > T)$. Encoding this property into an equation system yields the following:

$$\nu X(v{:}N) = (\forall m{:}N.\, X(v+m)) \wedge \forall n{:}N.\, v = n \implies n > T$$

The simple predicate $v > T$ is an invariant for equation X as it is easily seen to meet the invariance criterion, but it is clearly not a process invariant of M. □

6.3. Process invariants and process equivalences

As stated in the introduction, various process equivalences between LPEs have been encoded as PBES solution problems. In this section, we show that the notion of a process invariant gives rise to global invariants in the equation systems encoding all of the process equivalences of [4]. We consider each equivalence in isolation.

Throughout this section, we assume a specification S and an implementation M given by the following LPEs

$$M(d{:}D^M) = \sum \left\{ \sum_{e_{\mathsf{a}}:E^M_{\mathsf{a}}} c^M_{\mathsf{a}}(d,e) \implies \mathsf{a}(f^M_{\mathsf{a}}(d,e)) \cdot M(g^M_{\mathsf{a}}(d,e)) \mid \mathsf{a} \in \mathcal{A}ct \right\}$$

$$S(d{:}D^S) = \sum \left\{ \sum_{e_{\mathsf{a}}:E^S_{\mathsf{a}}} c^S_{\mathsf{a}}(d,e) \implies \mathsf{a}(f^S_{\mathsf{a}}(d,e)) \cdot S(g^S_{\mathsf{a}}(d,e)) \mid \mathsf{a} \in \mathcal{A}ct \right\}$$

The equation system that encodes strong bisimulation between M and S is given by Algorithm 1. Strong bisimulation is the finest equivalence that is still considered useful as a process equivalence.

Algorithm 1 Generation of a PBES sbisim encoding Strong Bisimilarity between LPEs M and S.

$\mathsf{sbisim} = \nu E$, **where**

$$\begin{aligned} E := \{ & X^{M,S}(d{:}D^M, d'{:}D^S) = \mathsf{match}^{M,S}(d,d') \wedge \mathsf{match}^{S,M}(d',d) \;, \\ & X^{S,M}(d'{:}D^S, d{:}D^M) = X^{M,S}(d,d') \;\} \end{aligned}$$

Where (For all $\mathsf{a} \in \mathcal{A}ct \wedge (p,q) \in \{(M,S),(S,M)\}$) we use the following abbreviations:

$$\mathsf{match}^{p,q}(d{:}D^p, d'{:}D^q) = \textstyle\bigwedge_{\mathsf{a}\in\mathcal{A}ct} \forall e{:}E^p_{\mathsf{a}}.(c^p_{\mathsf{a}}(d,e) \implies \mathsf{step}^{p,q}_{\mathsf{a}}(d,d',e));$$

$$\begin{aligned} & \mathsf{step}^{p,q}_{\mathsf{a}}(d{:}D^p, d'{:}D^q, e{:}E^p_{\mathsf{a}}) = \\ & \quad \exists e'{:}E^q_{\mathsf{a}}.c^q_{\mathsf{a}}(d',e') \wedge (f^p_{\mathsf{a}}(d,e) = f^q_{\mathsf{a}}(d',e')) \wedge X^{p,q}(g^p_{\mathsf{a}}(d,e), g^q_{\mathsf{a}}(d',e')); \end{aligned}$$

Theorem 53. *Let ι be a process invariant for LPE M. Let f^M be the simple function defined as:*

$$f^M(Z) = \iota \quad \text{for } Z \in \{X^{M,S}, X^{S,M}\}$$

Then f^M is a global invariant of sbisim.

Proof. Assume ι is a process invariant for the LPE M. We are required to show that:

$$\begin{aligned} (1) \qquad & (\iota \wedge X^{M,S}(d,d')) \\ \leftrightarrow\; & (\iota \wedge X^{M,S}(d,d')) \left[_{Z\in\{X^{M,S},X^{S,M}\}} (\iota \wedge Z(d_Z))_{\langle d_Z\rangle} / Z\right] \end{aligned}$$

(2) $(\iota \wedge \mathsf{match}^{M,S}(d,d') \wedge \mathsf{match}^{S,M}(d',d))$
$\leftrightarrow (\iota \wedge \mathsf{match}^{M,S}(d,d') \wedge \mathsf{match}^{S,M}(d',d)) \left[_{Z\in\{X^{M,S},X^{S,M}\}}(\iota \wedge Z(d_Z))_{\langle d_Z\rangle}/Z\right]$

Observe that the first requirement is readily satisfied. For the second requirement, we observe that:

$$
\begin{aligned}
&(\iota \wedge \mathsf{match}^{M,S}(d,d')) \left[_{Z\in\{X^{M,S},X^{S,M}\}}(\iota \wedge Z(d_Z))_{\langle d_Z\rangle}/Z\right]\\
\leftrightarrow\ &\{\text{By definition of syntactic substitution}\}\\
&\iota \wedge \bigwedge_{\mathsf{a}\in\mathcal{A}ct} \forall e{:}E^M_{\mathsf{a}}.\ (c^M_{\mathsf{a}}(d,e) \implies\\
&\quad \exists e'{:}E^S_{\mathsf{a}}.\ c^S_{\mathsf{a}}(d',e') \wedge f^M_{\mathsf{a}}(d,e) = f^S_{\mathsf{a}}(d',e')\\
&\qquad \wedge(\iota[g^M_{\mathsf{a}}(d,e)/d] \wedge X^{M,S}(g^M_{\mathsf{a}}(d,e), g^S_{\mathsf{a}}(d',e'))))\\
\leftrightarrow\ &\{\text{for all actions } \mathsf{a} \text{ and for all } e{:}E^M_{\mathsf{a}}\text{: } \iota \wedge c^M_{\mathsf{a}}(d,e) \text{ implies } \iota[g^M_{\mathsf{a}}(d,e)/d]\ \}\\
&\iota \wedge \bigwedge_{\mathsf{a}\in\mathcal{A}ct} \forall e{:}E^M_{\mathsf{a}}.\ (c^M_{\mathsf{a}}(d,e) \implies\\
&\quad \exists e'{:}E^S_{\mathsf{a}}.\ c^S_{\mathsf{a}}(d',e') \wedge f^M_{\mathsf{a}}(d,e) = f^S_{\mathsf{a}}(d',e') \wedge X^{M,S}(g^M_{\mathsf{a}}(d,e), g^S_{\mathsf{a}}(d',e')))\\
\leftrightarrow\ &\{\text{Definition of match}\}\\
&(\iota \wedge \mathsf{match}^{M,S}(d,d'))
\end{aligned}
$$

and, likewise:

$$
\begin{aligned}
&(\iota \wedge \mathsf{match}^{S,M}(d',d)) \left[_{Z\in\{X^{M,S},X^{S,M}\}}(\iota \wedge Z(d_Z))_{\langle d_Z\rangle}/Z\right]\\
\leftrightarrow\ &\{\text{By definition of syntactic substitution}\}\\
&\iota \wedge \bigwedge_{\mathsf{a}\in\mathcal{A}ct} \forall e{:}E^S_{\mathsf{a}}.\ \left(c^S_{\mathsf{a}}(d',e) \implies\right.\\
&\quad \exists e'{:}E^M_{\mathsf{a}}.\ c^M_{\mathsf{a}}(d,e') \wedge f^S_{\mathsf{a}}(d',e) = f^M_{\mathsf{a}}(d,e')\\
&\qquad \left.\wedge \left(\iota[g^M_{\mathsf{a}}(d,e')/d] \wedge X^{S,M}(g^S_{\mathsf{a}}(d',e), g^M_{\mathsf{a}}(d,e'))\right)\right)\\
\leftrightarrow\ &\{\text{For all actions } \mathsf{a} \text{ and for all } e'{:}E^M_{\mathsf{a}}\text{: } \iota \wedge c^M_{\mathsf{a}}(d,e') \text{ implies } \iota[g^M_{\mathsf{a}}(d,e')/d]\ \}\\
&\iota \wedge \bigwedge_{\mathsf{a}\in\mathcal{A}ct} \forall e{:}E^S_{\mathsf{a}}.\ (c^S_{\mathsf{a}}(d',e) \implies\\
&\quad \exists e'{:}E^M_{\mathsf{a}}.\ c^M_{\mathsf{a}}(d,e') \wedge f^S_{\mathsf{a}}(d',e) = f^M_{\mathsf{a}}(d,e') \wedge X^{S,M}(g^S_{\mathsf{a}}(d',e), g^M_{\mathsf{a}}(d,e')))\\
\leftrightarrow\ &\{\text{Definition of match}\}\\
&(\iota \wedge \mathsf{match}^{S,M}(d',d))
\end{aligned}
$$

From this, one can conclude that (2) holds, and, therefore that f^M is a global invariant. □

Branching bisimulation is an equivalence that allows one to use *abstraction* to relate two processes that *observationally* behave the same; moreover, it preserves the branching structure of processes, which is a distinguishing feature of the equivalence relation. The largest branching bisimulation relation that relates states of M to S and *vice versa*, can be found by solving the equation system that is generated by Algorithm 2, and therefore also the problem whether M is branching bisimilar to S for given initial states of M and S.

Theorem 54. *Let ι be a process invariant for LPE M. Let f^M be the simple function defined as:*

$$f^M(Z) = \begin{cases} \iota & \text{if } Z \in \{X^{M,S}, X^{S,M}, Y^{S,M}_{\mathsf{a}} \mid \mathsf{a} \in \mathcal{A}ct\}\\ \iota \wedge c^M_{\mathsf{a}}(d,e) & \text{if } Z \in \{Y^{M,S}_{\mathsf{a}} \mid \mathsf{a} \in \mathcal{A}ct\}\end{cases}$$

Then f^M is a global invariant of brbisim.

Proof. Let $V = \mathsf{bnd}(\mathsf{brbisim})$. Assume that ι is a process invariant for the LPE M. We have to show the following equivalences for the equations for $X^{M,S}$, $X^{S,M}$ and $Y^{M,S}_{\mathsf{a}}$, $Y^{S,M}_{\mathsf{a}}$:

(1) $\iota \wedge \mathsf{match}^{M,S}(d,d') \wedge \mathsf{match}^{S,M}(d',d)$
$\leftrightarrow (\iota \wedge \mathsf{match}^{M,S}(d,d') \wedge \mathsf{match}^{S,M}(d',d)) \left[_{Z\in V}(f^M(Z) \wedge Z(d_Z))_{\langle d_Z\rangle}/Z\right]$

(2) $\iota \wedge X^{M,S}(d,d')$
$\leftrightarrow (\iota \wedge X^{M,S}(d,d')) \left[_{Z\in V}(f^M(Z) \wedge Z(d_Z))_{\langle d_Z\rangle}/Z\right]$

(3) $c^M_{\mathsf{a}}(d,e) \wedge \iota \wedge \mathsf{close}^{M,S}_{\mathsf{a}}(d,d',e)$
$\leftrightarrow (c^M_{\mathsf{a}}(d,e) \wedge \iota \wedge \mathsf{close}^{M,S}_{\mathsf{a}}(d,d',e)) \left[_{Z\in V}(f^M(Z) \wedge Z(d_Z))_{\langle d_Z\rangle}/Z\right]$

Algorithm 2 Generation of a PBES brbisim encoding Branching Bisimilarity between LPEs M and S.

$$\begin{aligned}
&\mathsf{brbisim} = \nu E_2 \mu E_1, \textbf{ where}\\
&\quad E_2 := \{X^{M,S}(d:D^M, d':D^S) = \mathsf{match}^{M,S}(d,d') \wedge \mathsf{match}^{S,M}(d',d)\ ,\\
&\qquad\quad X^{S,M}(d':D^S, d:D^M) = X^{M,S}(d,d')\ \}\\
&\quad E_1 := \{Y^{p,q}_{\mathsf{a}}(d:D^p, d':D^q, e:E^p_{\mathsf{a}}) = \mathsf{close}^{p,q}_{\mathsf{a}}(d,d',e)\\
&\qquad\qquad \mid a \in \mathcal{A}ct \wedge (p,q) \in \{(M,S),(S,M)\}\}
\end{aligned}$$

Where (for all $\mathsf{a} \in \mathcal{A}ct$, $(p,q) \in \{(M,S),(S,M)\}$) we use the following abbreviations:

$$\mathsf{match}^{p,q}(d:D^p, d':D^q) = \textstyle\bigwedge_{\mathsf{a}\in\mathcal{A}ct} \forall e{:}E^p_{\mathsf{a}}.\ (c^p_{\mathsf{a}}(d,e) \implies Y^{p,q}_{\mathsf{a}}(d,d',e));$$

$$\begin{aligned}
\mathsf{close}^{p,q}_{\mathsf{a}}(d:D^p, d':D^q, e:E^p_{\mathsf{a}}) = {}& \exists e'{:}E^q_\tau.\ (c^q_\tau(d',e') \wedge Y^{p,q}_{\mathsf{a}}(d, g^q_\tau(d',e'),e))\\
&\vee (X^{p,q}(d,d') \wedge \mathsf{step}^{p,q}_{\mathsf{a}}(d,d',e));
\end{aligned}$$

$$\begin{aligned}
\mathsf{step}^{p,q}_{\mathsf{a}}(d:D^p, d':D^q, e:E^p_{\mathsf{a}}) = {}& (\mathsf{a} = \tau \wedge X^{p,q}(g^p_\tau(d,e), d'))\vee\\
&\exists e'{:}E^q_{\mathsf{a}}.\ c^q_{\mathsf{a}}(d',e') \wedge (f^p_{\mathsf{a}}(d,e) = f^q_{\mathsf{a}}(d',e'))\wedge\\
&X^{p,q}(g^p_{\mathsf{a}}(d,e), g^q_{\mathsf{a}}(d',e'));
\end{aligned}$$

(4) $\quad \iota \wedge \mathsf{close}^{S,M}_{\mathsf{a}}(d,d',e)$

$\leftrightarrow\ (\iota \wedge \mathsf{close}^{S,M}_{\mathsf{a}}(d,d',e))\ \left[_{Z\in V}(f^M(Z) \wedge Z(d_Z))_{\langle d_Z\rangle}/Z\right]$

For the first equivalence, we reason as follows:

$$\begin{aligned}
&(\iota \wedge \mathsf{match}^{M,S}(d,d') \wedge \mathsf{match}^{S,M}(d',d))\ \left[_{Z\in V}(f^M(Z) \wedge Z(d_Z))_{\langle d_Z\rangle}/Z\right]\\
\leftrightarrow\ & \{\text{Definition of match and syntactic substitution}\}\\
&\iota \wedge \textstyle\bigwedge_{\mathsf{a}\in\mathcal{A}ct} \forall e{:}E^M_{\mathsf{a}}.\ c^M_{\mathsf{a}}(d,e) \implies (c^M_{\mathsf{a}}(d,e) \wedge \iota \wedge Y^{M,S}_{\mathsf{a}}(d,d',e))\\
&\quad \textstyle\bigwedge_{\mathsf{a}\in\mathcal{A}ct} \forall e{:}E^M_{\mathsf{a}}.\ c^S_{\mathsf{a}}(d',e) \implies (\iota \wedge Y^{M,S}_{\mathsf{a}}(d,d',e))\\
\leftrightarrow\ & \{\text{Logic}\}\\
&\iota \wedge \textstyle\bigwedge_{\mathsf{a}\in\mathcal{A}ct} \forall e{:}E^M_{\mathsf{a}}.\ c^M_{\mathsf{a}}(d,e) \implies Y^{M,S}_{\mathsf{a}}(d,d',e)\\
&\quad \textstyle\bigwedge_{\mathsf{a}\in\mathcal{A}ct} \forall e{:}E^M_{\mathsf{a}}.\ c^S_{\mathsf{a}}(d',e) \implies Y^{M,S}_{\mathsf{a}}(d,d',e)\\
\leftrightarrow\ & \{\text{Definition of match}\}\\
&(\iota \wedge \mathsf{match}^{M,S}(d,d') \wedge \mathsf{match}^{S,M}(d',d))
\end{aligned}$$

Equivalence (2) follows immediately from the definition of syntactic substitution and idempotence of $\wedge$. We next focus on equivalence (3):

$$\begin{aligned}
&(c^M_{\mathsf{a}}(d,e) \wedge \iota \wedge \mathsf{close}^{M,S}_{\mathsf{a}}(d,d',e))\ \left[_{Z\in V}(f^M(Z) \wedge Z(d_Z))_{\langle d_Z\rangle}/Z\right]\\
\leftrightarrow\ & \{\text{Definition of close and syntactic substitution }\}\\
&(c^M_{\mathsf{a}}(d,e) \wedge \iota \wedge (\exists e'{:}E^S_\tau.\ (c^S_\tau(d',e') \wedge (c^M_{\mathsf{a}}(d,e) \wedge \iota \wedge Y^{M,S}_{\mathsf{a}}(d, g^S_\tau(d',e'),e)))\\
&\quad \vee((\iota \wedge X^{M,S}(d,d')) \wedge ((\mathsf{a} = \tau \wedge \iota[g^M_\tau(d,e)/d] \wedge X^{M,S}(g^M_\tau(d,e),d'))\\
&\qquad \vee \exists e'{:}E^S_{\mathsf{a}}.\ c^S_{\mathsf{a}}(d',e') \wedge f^M_{\mathsf{a}}(d,e) = f^S_{\mathsf{a}}(d',e')\\
&\qquad\quad \wedge(\iota[g^M_{\mathsf{a}}(d,e)/d] \wedge X^{p,q}(g^M_{\mathsf{a}}(d,e), g^S_{\mathsf{a}}(d',e'))))))\\
\leftrightarrow\ & \{c^M_{\mathsf{a}}(d,e) \wedge \iota \text{ implies } \iota[g^M_{\mathsf{a}}(d,e)/d] \text{ (likewise for } \mathsf{a} = \tau\text{); logic}\}\\
&(c^M_{\mathsf{a}}(d,e) \wedge \iota \wedge (\exists e'{:}E^S_\tau.\ (c^S_\tau(d',e') \wedge Y^{M,S}_{\mathsf{a}}(d, g^S_\tau(d',e'),e))\\
&\quad \vee(X^{M,S}(d,d') \wedge ((\mathsf{a} = \tau \wedge X^{M,S}(g^M_\tau(d,e),d'))\\
&\qquad \vee \exists e'{:}E^S_{\mathsf{a}}.\ c^S_{\mathsf{a}}(d',e') \wedge f^M_{\mathsf{a}}(d,e) = f^S_{\mathsf{a}}(d',e')\\
&\qquad\quad \wedge X^{p,q}(g^M_{\mathsf{a}}(d,e), g^S_{\mathsf{a}}(d',e'))))))\\
\leftrightarrow\ & \{\text{Definition of close}\}\\
&c^M_{\mathsf{a}}(d,e) \wedge \iota \wedge \mathsf{close}^{M,S}_{\mathsf{a}}(d,d',e)
\end{aligned}$$

Equivalence (4) is a variation of the above reasoning and is therefore omitted. Note that the slightly weaker invariant for $Y^{S,M}_{\mathsf{a}}$ is due to the fact that each occurrence of $Y^{S,M}_{\mathsf{a}}$ in $\mathsf{close}^{S,M}_{\mathsf{a}}$ is under the scope of a predicate $c^M_{\mathsf{a}}(d,e')$, which is not the case for $Y^{M,S}_{\mathsf{a}}$. □

Branching simulation equivalence is a weaker equivalence than branching bisimulation, but follows roughly the same translation (see Algorithm 3).

Algorithm 3 Generation of a PBES brsim encoding (Branching) Simulation Equivalence between LPEs M and S.

$\mathsf{brsim}(m, n) = \nu E_2 \mu E_1$, **where**

$$E_2 := \{X(d{:}D^M, d'{:}D^S) = X^{M,S}(d, d') \wedge X^{S,M}(d', d),$$
$$X^{M,S}(d{:}D^M, d'{:}D^S) = \mathsf{match}^{M,S}(d, d'),$$
$$X^{S,M}(d'{:}D^S, d{:}D^M) = \mathsf{match}^{S,M}(d', d)\}$$
$$E_1 := \{Y^{p,q}_{\mathsf{a}}(d{:}D^p, d'{:}D^q, e{:}E^p_{\mathsf{a}}) = \mathsf{close}^{p,q}_{\mathsf{a}}(d, d', e) \mid \mathsf{a} \in \mathcal{A}ct \wedge (p, q) \in \{(M, S), (S, M)\}\}$$

Where match and close are as defined in Algorithm 2.

Theorem 55. *Let ι be a process invariant for LPE M. Let f^M be the simple function defined as:*

$$f^M(Z) = \begin{cases} \iota & \text{if } Z \in \{X, X^{M,S}, X^{S,M}, Y^{S,M}_{\mathsf{a}} \mid \mathsf{a} \in \mathcal{A}ct\} \\ \iota \wedge c^M_{\mathsf{a}}(d, e) & \text{if } Z \in \{Y^{M,S}_{\mathsf{a}} \mid \mathsf{a} \in \mathcal{A}ct\} \end{cases}$$

Then f^M is a global invariant of brsim.

Proof. The proof follows the same line of reasoning as the proof for Theorem 54, and is therefore omitted. □

Algorithm 4 Generation of a PBES wbisim encoding Weak Bisimilarity between LPEs M and S.

$\mathsf{wbisim} = \nu E_2 \mu E_1$, **where**

$$E_2 := \{X^{M,S}(d{:}D^M, d'{:}D^S) = \mathsf{match}^{M,S}(d, d') \wedge \mathsf{match}^{S,M}(d', d)\ ,$$
$$X^{S,M}(d'{:}D^S, d{:}D^M) = X^{M,S}(d, d')\ \}$$
$$E_1 := \{Y^{p,q}_{1,\mathsf{a}}(d{:}D^p, d'{:}D^q, e{:}E^p_{\mathsf{a}}) = \mathsf{close}^{p,q}_{1,\mathsf{a}}(d, d', e),$$
$$Y^{p,q}_{2,\mathsf{a}}(d{:}D^p, d'{:}D^q) = \mathsf{close}^{p,q}_{2,\mathsf{a}}(d, d'), \mid \mathsf{a} \in \mathcal{A}ct \wedge (p, q) \in \{(M, S), (S, M)\}\}$$

Where (for all $\mathsf{a} \in \mathcal{A}ct \wedge (p, q) \in \{(M, S), (S, M)\}$) we use the following abbreviations:

$$\mathsf{match}^{p,q}(d{:}D^p, d'{:}D^q) = \bigwedge_{\mathsf{a}\in\mathcal{A}ct} \forall e{:}E^p_{\mathsf{a}}.(c^p_{\mathsf{a}}(d, e) \implies Y^{p,q}_{1,\mathsf{a}}(d, d', e));$$

$$\mathsf{close}^{p,q}_{1,\mathsf{a}}(d{:}D^p, d'{:}D^q, e{:}E^p_{\mathsf{a}}) = \exists e'{:}E^q_\tau.(c^q_\tau(d', e') \wedge Y^{p,q}_{1,\mathsf{a}}(d, g^q_\tau(d', e'), e)) \vee \mathsf{step}^{p,q}_{\mathsf{a}}(d, d', e);$$

$$\mathsf{step}^{p,q}_{\mathsf{a}}(d{:}D^p, d'{:}D^q, e{:}E^p_{\mathsf{a}}) = (\mathsf{a} = \tau \wedge Y^{p,q}_{2,\mathsf{a}}(g^p_{\mathsf{a}}(d, e), d')) \vee \exists e'{:}E^q_{\mathsf{a}}.c^q_{\mathsf{a}}(d', e') \wedge (f^p_{\mathsf{a}}(d, e) = f^q_{\mathsf{a}}(d', e')) \wedge Y^{p,q}_{2,\mathsf{a}}(g^p_{\mathsf{a}}(d, e), g^q_{\mathsf{a}}(d', e'));$$

$$\mathsf{close}^{p,q}_{2,\mathsf{a}}(d{:}D^p, d'{:}D^q) = X^{p,q}(d, d') \vee \exists e'{:}E^q_\tau.c^q_\tau(d', e') \wedge Y^{p,q}_{2,\mathsf{a}}(d, g^q_\tau(d', e'));$$

Theorem 56. *Let ι be a process invariant for LPE M. Let f^M be the simple function defined as:*

$$f^M(Z) = \begin{cases} \iota & \text{if } Z \in \{X^{M,S}, X^{S,M}, Y^{S,M}_{1,\mathsf{a}}, Y^{S,M}_{2,\mathsf{a}}, Y^{M,S}_{2,\mathsf{a}} \mid \mathsf{a} \in \mathcal{A}ct\} \\ \iota \wedge c^M_{\mathsf{a}}(d, e) & \text{if } Z \in \{Y^{M,S}_{1,\mathsf{a}} \mid \mathsf{a} \in \mathcal{A}ct\} \end{cases}$$

Then f^M is a global invariant of wbisim.

Proof. The proof follows the same line of reasoning as the proof for Theorems 53 and 54. □

A benefit of using process invariants in solving equation systems encoding a particular equivalence is that, when using a symbolic approximation, unreachable parts of the processes involved are removed from the approximants by means of the invariants. In general, this greatly accelerates the approximation process.

7. Examples

So far, we have studied the theory of invariants in equation systems. We next illustrate how invariants can help solving equation systems that arise from a diversity of verification problems, such as model checking and equivalence checking. In order to focus on the notion of invariance and its accompanying theory in equation systems, we mostly focus on examples that are not overly complex, but that cannot be solved straightforwardly within the theory of equation systems without the use of invariants. The last example concerns the verification of a Cache Coherence Protocol, which illustrates that equation system invariants can also be used in non-academic verification problems.

7.1. Insertion sort

Insertion sort [17] is a classical sorting algorithm, in which the sorted list is built one entry at a time. In the variant discussed here, the algorithm starts by sorting the last two elements of the input list. In every further iteration, the last element of the unsorted part of the list is inserted into its correct position in the already-sorted part. To prove its correctness in the PBES framework, we use invariants that are already known [17] and which, it turns out, can immediately be checked on the process model and used as PBES invariants as ensured by Theorem 51. The example also demonstrates the use of more complex invariants involving quantifiers. We can model insertion sort as a process with parameter l, which is of sort list of naturals ($\mathcal{L}(N)$) and parameters i, j, key. Index j keeps track of the algorithm's iterations. In every iteration, a search for the correct position of $l.j$ is started by action `search` and pursued by comparing all elements with index $i < j$ to key, which has been set to $l.j$. As long as $l.i > key$, $l.i$ is moved to position $i + 1$ by action `move`. When the correct position is found, $l.j$ gets inserted using action `insert`. We write $l[e/i]$ to denote the list l in which position i is assigned value e, i.e. $l.i := e$, and the remainder of the list is unmodified.

$$
\begin{aligned}
I(l{:}\mathcal{L}(N), j, i, key{:}N) = {} & (j \geq |l|) \implies \texttt{stop}(l) \cdot \delta \\
& + (j < |l| \wedge i = j) \implies \texttt{search} \cdot I(l, j, j-1, l.j) \\
& + (0 < i < j < |l| \wedge l.i > key) \implies \texttt{move} \cdot I(l[l.i/i+1], j, i-1, key) \\
& + (j < |l| \wedge (i = 0 \vee l.i \leq key)) \implies \texttt{insert} \cdot I(l[key/i+1], j+1, j+1, 0)
\end{aligned}
$$

Upon execution of the action `stop`, the algorithm terminates, leading to a deadlock state (modelled by δ), i.e., a state that has no enabled actions. Partial correctness of the algorithm means that the list that is reported via action `stop` is sorted:

$$\nu X.\ [\top]X \wedge \forall l'{:}\mathcal{L}(N).\ [\texttt{stop}(l')](\forall n{:}N.\ n < |l'| \implies l'.n \leq l'.n+1) \qquad (\phi)$$

Note that this property does not require the input and output lists to be related, nor that the process is guaranteed to stop. Encoding the model checking problem $I(l, j, i, key) \models \phi$ yields the following equation system:

$$
\begin{aligned}
& \nu X(l{:}\mathcal{L}(N), j, i, key{:}N) = \\
& \quad ((j < |l| \wedge i = j) \implies X(l, j, j-1, l.j)) \\
& \wedge (0 < i < j < |l| \wedge l.i > key) \implies X(l[l.i/i+1], j, i-1, key) \\
& \wedge (j < |l| \wedge (i = 0 \vee l.i \leq key)) \implies X(l[key/i+1], j+1, j+1, 0) \\
& \wedge \forall l'{:}\mathcal{L}(N).\ j \geq |l| \wedge l' = l \implies (\forall n.\ n < |l'| \implies l.n \leq l.n+1)
\end{aligned}
$$

Note that we have the following equivalence:

$$
\begin{aligned}
& (\forall l'{:}\mathcal{L}(N).\ j \geq |l| \wedge l' = l \implies (\forall n.\ n < |l'| \implies l.n \leq l.n+1)) \\
\leftrightarrow\ & (j \geq |l| \implies (\forall n.\ n < |l| \implies l.n \leq l.n+1))
\end{aligned}
$$

which can be used to further reduce the complexity of the above equation system. A direct symbolic approximation for the resulting equation system, however, still does not converge.

Next, we demonstrate that a suitable invariant immediately leads to a solution to the original problem. For this, we get inspiration from the standard invariant for insertion sort, which is used in the axiomatic semantics to prove correctness of the algorithm. Our invariant combines an invariant for the j's loop and an invariant for the i's loop. The first one states that the list is sorted up to position $j - 1$. The second one states that all elements of l between indices i and j are larger than the key key, and that, during the search phase, the upper end of the current sublist is also sorted, i.e. $l.(j-1) < l.j$.

$$
\begin{aligned}
& \forall n{:}N.\ (1 \leq n < j-1 \implies l.n \leq l.(n+1)) \\
\wedge\ & \forall n{:}N.\ (i < n < j \implies l.n \geq key) \\
\wedge\ & (i < j-1 \implies l.(j-1) \leq l.j).
\end{aligned}
$$

The above formula can be checked to be an invariant of process I, and, by Theorem 51, it is therefore also an invariant for equation X. Moreover, we find:

$$
\begin{aligned}
& \forall n{:}N.\ (1 \leq n \leq j \implies l.n \leq l.n+1) \wedge \forall n{:}N.\ (i < n \leq j \implies l.n \geq key) \\
\rightarrow\ & (j \geq |l| \implies (\forall n.\ n < |l| \implies l.n \leq l.n+1))
\end{aligned}
$$

As a consequence, the invariant-strengthened equation system can be brought into the form of Proposition 38, from which it follows that the invariant is the solution to the invariant-strengthened equation system. This means that for any system $I(l, i, j, key)$ with values for i, j and key, and any list l that satisfy the invariant, property ϕ holds.

7.2. A simple voting protocol

Electronic voting protocols like [8] are usually rather complex. They employ data encryption techniques and several central entities, together which guarantee the correctness of the outcome and simultaneously protect the privacy of the voters against various (coalitions of) internal and external parties. For the illustrative purpose of this section, we consider a very basic electronic referendum and use the PBES theory to analyse whether the privacy of the voters is guaranteed with respect to external observers.

Our model is given by the LPE E, where the intended votes of participants are modelled by variable V, which is of sort $\mathcal{L}(\{0,1\})$ (list of bits). We write $V.i$ to indicate the vote of voter i; $V.i = 1$ if voter i votes *yes* and 0 for *no*. The set of registered voters is kept as R and y, n are the number of positive and negative casted votes, respectively. The act of voting of a single person is modelled by action $\mathrm{v}(i)$ with $i \in R$, and voting proceeds in random order. The actual yes/no vote increases the corresponding counter. When all registered participants have voted, the outcome is published.

$$\begin{aligned} &E(V{:}\mathcal{L}(\{0,1\}), R{:}2^N, y, n{:}N) = \\ &\quad (R = \emptyset) \Longrightarrow \mathtt{outcome}(y,n) \cdot \delta \\ &\quad + \textstyle\sum_{i:N}\ i \in R \Longrightarrow \mathtt{vote}(i) \cdot E(V, R \setminus \{i\}, y + V.i, n + (1 - V.i)) \end{aligned}$$

One way to formalise privacy of the voting process is as the inability of an external observer to tell whether $V.i = 0$ or $V.i = 1$ for any voter i. In other words, privacy is guaranteed if process $E(l, r, 0, 0)$ is strongly bisimilar to $E(\pi(l), r, 0, 0)$, where list $\pi(l)$ is a permutation of list l and $l.i$ ($\pi(l).i$, resp.) is defined for every voter $i \in r$. Strong bisimilarity can be encoded using an equation system, see below for the resulting equation system (after some minor logical rewriting); for the general encoding, see the preceding section.

$$\begin{aligned} &(\nu X(V{:}\mathcal{L}(\{0,1\}), R{:}2^N, y, n{:}N, V'{:}\mathcal{L}(\{0,1\}), R'{:}2^N, y', n'{:}N) = \\ &(\forall i{:}N.\ i \in R \implies (i \in R' \\ &\wedge X(V, R \setminus \{i\}, y + V.i, n + (1 - V.i), V', R' \setminus \{i\}, y' + V'.i, n' + (1 - V'.i)))) \\ &\wedge (\forall i{:}N.\ i \in R' \Rightarrow (i \in R \\ &\wedge X'(V, R \setminus \{i\}, y + V.i, n + (1 - V.i), V', R' \setminus \{i\}, y' + V'.i, n' + (1 - V'.i)))) \\ &\wedge (R = \emptyset \Longleftrightarrow R' = \emptyset) \wedge (R = \emptyset \implies (y = y' \wedge n = n'))) \\ &(\nu X'(V'{:}\mathcal{L}(\{0,1\}), R'{:}2^N, y', n'{:}N, V{:}\mathcal{L}(\{0,1\}), R{:}2^N, y, n{:}N) = \\ &X(V, R, y, n, V', R', y', n')) \end{aligned}$$

All occurrences of predicate variable X' in the equation for X can be removed by a substitution. A subsequent standard symbolic approximation of variable X generates a series of increasingly complex equations expressing constraints on subsets of R and does not converge.

Instead, we use an invariant to simplify matters. The equation system encodes the situation when two arbitrary processes E are strongly bisimilar, i.e. regardless of the initial states of the LPEs. This is an additional source of complexity. We, on the other hand are interested only in solving the equation system for lists V and V' that are permutations of one another and sets of voters that are the same. We state the following three simple predicate formulae:

- ι_1, defined as $R = R'$ formalises that we are not interested in relating information for different sets of voters,
- ι_2, defined as $y + n = y' + n'$ formalises that the total number of expressed votes should be the same in both protocols,
- ι_3, given by $y + \sum_{i \in R} V.i = y' + \sum_{i \in R'} V'.i$ formalises that V and V' are permutations of one another.

Let ι be the predicate formula $\iota_1 \wedge \iota_2 \wedge \iota_3$; ι is an invariant for X and X', since it satisfies the sufficiency criteria from Property 28, so we may use this invariant to strengthen the equation system. We furthermore observe that for equation X:

$$\iota \rightarrow (R = \emptyset \Longleftrightarrow R' = \emptyset) \wedge (R = \emptyset \implies (y = y' \wedge n = n'))$$

From this it follows that:

$$\iota \Longleftrightarrow (\iota \wedge (R = \emptyset \Longleftrightarrow R' = \emptyset) \wedge (R = \emptyset \implies (y = y' \wedge n = n')))$$

This means that, after a substitution step which removes X' from the equation for X, the strengthened equation for X is of the form of Proposition 38, from which it immediately follows that it has solution ι. Since the following is a tautology:

$$\iota[l/V, r/R, 0/y, 0/n, \pi(l)/V', r/R', 0/y', 0/n']$$

we find that $E(l, r, 0, 0)$ and $E(\pi(l), r, 0, 0)$ are strongly bisimilar and privacy is therefore guaranteed.

7.3. Infinite buffer

In this section, we demonstrate the use of an invariant for process verification, in which an invariant of the PBES encoding the verification problem is essential for solving the problem. The invariant is a property that relates parameters that are due to the process and parameters that are due to the modal formula. The system is a simple unbounded buffer, on which two operations are possible: reading and writing to the buffer:

$$\begin{aligned} B(q{:}\mathcal{Q}) \ = &\sum_{m:M} \mathtt{r}(m) \cdot B([m] +\!\!+ q) \\ &+ |q| > 0 \Longrightarrow \mathtt{w}(\mathsf{head}(q)) \cdot B(\mathsf{tail}(q)) \end{aligned}$$

The sort $\mathcal{Q}$ is the sort of queues containing messages of sort M. The head of a queue q is denoted by $\mathsf{head}(q)$ and, likewise, the tail of q is denoted $\mathsf{tail}(q)$. The property that we wish to verify is the following: writing a message m to the buffer via action

$\mathtt{r}$ eventually leads to sending the message via action $\mathtt{w}$ along all fair paths of the buffer. This is captured by the following μ-calculus formula:

$$\nu X.[\top]X \wedge \forall m{:}M.\ ([\mathtt{r}(m)]\nu Y.[\neg\mathtt{w}(m)]Y \wedge \mu Z.(\langle\neg\mathtt{w}(m)\rangle Z \vee \langle\mathtt{w}(m)\rangle\top))$$

Encoding the satisfaction of this property by process $B(q)$ in a PBES, and subsequent logical simplification of the predicate formulae, gives rise to the following equation system:

$$\begin{aligned}
&(\nu X(q{:}\mathcal{Q}) = (\forall m{:}M.\ X([m] +\!\!+ q) \wedge Y([m] +\!\!+ q, m))\\
&\qquad \wedge(|q| > 0 \implies X(\mathsf{tail}(q))))\\
&(\nu Y(q{:}\mathcal{Q}, m{:}M) = \forall m'{:}M.\ Y([m'] +\!\!+ q, m)\\
&\qquad \wedge(|q| > 0 \wedge \mathsf{head}(q) \neq m \implies Y(\mathsf{tail}(q), m)) \wedge Z(q, m))\\
&(\mu Z(q{:}\mathcal{Q}, m{:}M) = (\exists m'{:}M.\ Z([m'] +\!\!+ q)\\
&\qquad \vee((|q| > 0 \wedge (\mathsf{head}(q) = m \vee Z(\mathsf{tail}(q), m))))))
\end{aligned}$$

A global invariant for the above equation system is $\top$ for X and $m \in q$ (meaning that m occurs somewhere in the queue q) for Y and Z. From hereon, assume that all equations have been strengthened with their local invariants. The invariant does not help in calculating the solution to Z; however, Z can be strengthened [15] to the following equation, leading to a possible under-approximation of the other predicate variables (so we may falsely conclude that X and Y are not true, whereas they would be true without the under-approximation):

$$(\mu Z(q{:}\mathcal{Q}, m{:}M) = m \in q \wedge (\mathsf{head}(q) = m \vee Z(\mathsf{tail}(q), m)))$$

The solution to this equation is obtained by a simple pattern matching [15], which, after some basic rewriting, yields:

$$(\mu Z(q{:}\mathcal{Q}, m{:}M) = m \in q)$$

A subsequent substitution of this solution in an invariant-strengthened equation for Y yields the following equivalent equation for Y:

$$\begin{aligned}
&(\nu Y(q{:}\mathcal{Q}, m{:}M) = m \in q \wedge \forall m'{:}M.\ Y([m'] +\!\!+ q, m)\\
&\quad \wedge(|q| > 0 \wedge \mathsf{head}(q) \neq m \implies Y(\mathsf{tail}(q), m)) \wedge m \in q)
\end{aligned}$$

Since $m \in q$ is a global invariant, we find that, as a consequence of Proposition 38, the solution to the above equation is $m \in q$. A final substitution of this solution for Y in the equation for X leads to the following equivalent equation for X:

$$(\nu X(q{:}\mathcal{Q}) = (\forall m{:}M.\ X([m] +\!\!+ q)) \wedge (|q| > 0 \implies X(\mathsf{tail}(q))))$$

Again, a symbolic approximation of this equation system immediately leads to the following equivalent equation for X:

$$(\nu X(q{:}\mathcal{Q}) = \top)$$

Since the process $B(q)$ satisfies the μ-calculus formula iff $X(q)$ is true, we can conclude that the property is indeed satisfied.

7.4. Readers–writers mutual exclusion

We consider a mutual exclusion problem between distributed clients. The process serves two types of clients, viz. *readers* and *writers*. It has to ensure that when one client writes, no other client may read or write, but several clients may read at the same time. A total of $N > 0$ readers and writers are assumed.

$$\begin{aligned}
P(n_r, n_w, t{:}N) \ = \ & t \geq 1 \implies \mathtt{r_s} \cdot P(n_r + 1, n_w, t - 1)\\
& + n_r > 0 \implies \mathtt{r_e} \cdot P(n_r - 1, n_w, t + 1)\\
& + t \geq N \implies \mathtt{w_s} \cdot P(n_r, n_w + 1, t - N)\\
& + n_w > 0 \implies \mathtt{w_e} \cdot P(n_r, n_w - 1, t + N)
\end{aligned}$$

Here the actions $\mathtt{r_s}$ and $\mathtt{w_s}$ express the starting of reading and writing of a client, respectively. Likewise, the actions $\mathtt{r_e}$ and $\mathtt{w_e}$ express the ending of reading and writing of a client, respectively. The variables n_r and n_w represent the number of active readers and active writers. The role of the variable t is to ensure the mutual exclusion property, by recording the weights of the number of active readers (weight 1) and writers (weight N).

The mutual exclusion problem (1). A property expressing that the above process indeed guarantees mutual exclusion between readers and writers follows from the following two properties:

(1) No writer can start if readers are reading: $\nu X.[\top]X \wedge [\mathtt{r_s}]\nu Y.([\neg \mathtt{r_e}]Y \wedge [\mathtt{w_s}]\bot)$.
(2) No reader can start if writers are busy: $\nu X.[\top]X \wedge [\mathtt{w_s}]\nu Y.([\neg \mathtt{w_e}]Y \wedge [\mathtt{r_s}]\bot)$.

We only treat the first property, as the second property goes along the same lines. The equation system that encodes the first property is, after some simplification, given below:

$$\begin{array}{l}(\nu X(n_r, n_w, t{:}N) = ((t \geq 1 \implies (X(n_r + 1, n_w, t - 1) \wedge Y(n_r + 1, n_w, t - 1)))\\ \quad \wedge (n_r > 0 \implies X(n_r - 1, n_w, t + 1)) \wedge (t \geq N \implies X(n_r, n_w + 1, t - N))\\ \qquad \wedge (n_w > 0 \implies X(n_r, n_w - 1, t + N))))\\ (\nu Y(n_r, n_w, t{:}N) = (t < N \wedge (t \geq 1 \implies Y(n_r + 1, n_w, t - 1))\\ \quad \wedge (n_w > 0 \implies Y(n_r, n_w - 1, t + N))))\end{array}$$

With standard techniques, Y can only be solved using an unwieldy pattern [15], which introduces multiple quantifications and additional selector functions; symbolic approximation does not converge in a finite number of steps. The use of invariants is the most appropriate strategy here. An invariant of process P is $t = N - (n_r + n_w \cdot N)$, which, by Theorem 51 is also a global invariant for the equations X and Y. Furthermore, $n_r \geq 1$ for Y and $\top$ for X is a global invariant. Both X and Y can be strengthened with the above invariants. The simple predicate formula $t < N$ follows from $t = N - (n_r + n_w \cdot N) \wedge n_r \geq 1$. We can employ Proposition 38 and simplify the equation for Y to the one below:

$$(\nu Y(n_r, n_w, t{:}N) = t = N - (n_r + n_w \cdot N))$$

Substituting this solution for Y in X and using Proposition 25 to simplify the resulting equation, we find the following equivalent equation for X:

$$\begin{array}{l}(\nu X(n_r, n_w, t{:}N) = ((t \geq 1 \implies (X(n_r + 1, n_w, t - 1)))\\ \quad \wedge (n_r > 0 \implies X(n_r - 1, n_w, t + 1)) \wedge (t \geq N \implies X(n_r, n_w + 1, t - N))\\ \qquad \wedge (n_w > 0 \implies X(n_r, n_w - 1, t + N))\\ \qquad\quad \wedge t = N - (n_r + n_w \cdot N)))\end{array}$$

Another application of Proposition 38 immediately leads to the solution $t = N - (n_r + n_w \cdot N)$ for X. Thus, writers cannot start writing while readers are active if initially the values for n_r, n_w, t satisfy $t = N - (n_r + n_w \cdot N)$.

The mutual exclusion problem (2). Alternatively, mutual exclusion can be checked by keeping track of the number of readers and writers active at any moment. One possibility is to "record" this information using dedicated data variables r and w in the μ-calculus formula. Using these data variables, we check for the following property: $w + r > 0 \implies r \cdot w = 0$ (i.e. if a reader is active, then no writer is active, and *vice versa*). This is achieved by the following μ-calculus formula, which states that this property holds if initially no readers and writers are active and regardless of the actions undertaken by the system:

$$\begin{array}{l}\nu X(r, w{:}N := 0, 0).\ (r + w > 0 \implies r \cdot w = 0)\\ \quad \wedge [\mathtt{r_s}]X(r + 1, w) \wedge [\mathtt{r_e}]X(r - 1, w) \wedge [\mathtt{w_s}]X(r, w + 1) \wedge [\mathtt{w_e}]X(r, w - 1)\end{array}$$

The equation system encoding the above property is the following:

$$\begin{array}{l}(\nu X(n_r, n_w, t, r, w{:}N) = (r + w > 0 \implies r \cdot w = 0)\\ \quad \wedge (t \geq 1 \implies X(n_r + 1, n_w, t - 1, r + 1, w))\\ \quad \wedge (n_r > 0 \implies X(n_r - 1, n_w, t + 1, r - 1, w))\\ \quad \wedge (t \geq N \implies X(n_r, n_w + 1, t - N, r, w + 1))\\ \quad \wedge (n_w > 0 \implies X(n_r, n_w - 1, t + N, r, w - 1)))\end{array}$$

Using Property 28, $r = n_r$ and $w = n_w$ immediately follow as invariants; these allow us to remove parameters n_r and n_w and replace all occurrences of n_r by r and n_w by w, respectively. We can furthermore prove that $t = N - (r + w \cdot N)$ and $r + w > 0 \implies r \cdot w = 0$ are invariants, which allows us to simplify the invariant-strengthened equation to:

$$\begin{array}{l}(\nu X(t, r, w{:}N) = (r + w > 0 \implies r \cdot w = 0) \wedge t = N - (r + w \cdot N)\\ \quad \wedge (t \geq 1 \implies X(t - 1, r + 1, w)) \wedge (r > 0 \implies X(t + 1, r - 1, w))\\ \qquad \wedge (t \geq N \implies X(t - N, r, w + 1)) \wedge (w > 0 \implies X(t + N, r, w - 1)))\end{array}$$

Proposition 38 allows us to conclude that the property at least holds if initially $n_r = r\ (= 0$, which is required by the μ-calculus formula), $n_w = w (= 0)$ and $t = N$. Note that this is a more restrictive result than the one obtained previously, but this is mainly due to the stricter formula that is checked; additional quantifiers may weaken the μ-calculus formula.

7.5. Mutual exclusion in a token ring

Let us consider the following algorithm executed by a set of processes arranged in a ring communication topology:

$$\begin{aligned} TR(l{:}\mathcal{L}(\{0,1,2\}), t{:}N) \;=\; & (l.t = 0) \implies \mathtt{pass}(t).TR(l, \mathsf{next}(t)) \\ & + (l.t = 0) \implies \mathtt{pass}(t).TR(l[1/t], \mathsf{next}(t)) \\ & + (l.t = 1) \implies \mathtt{enter}(t).TR(l[2/t], t) \\ & + (l.t = 2) \implies \mathtt{idle}(t).TR(l[0/t], t) \end{aligned}$$

The list l keeps track of the current state of all processes; $l.i$ indicates the current state of process i, which can be 0 (idle), 1 (waiting) or 2 (critical section). Parameter t is the index of the current process holding the token. Upon receiving the token, a process in location 0 passes it on to the next process in the ring and may move to location 1 or stay at 0. If the token process is in location 1, it will enter its critical section, and if in location 2, it will exit it. The function $\mathsf{next}(t)$ computes the next process that receives the token; the analysis below is valid with respect to any implementation for this function.

We are interested in the mutual exclusion property, which says that no two processes are simultaneously in their critical section. Formally, whenever an action $\mathtt{enter}(i)$ has taken place, $\mathtt{idle}(i)$ should also take place, before any other action $\mathtt{enter}(j)$ gets enabled:

$$\begin{aligned} & \nu X.\ (\forall i{:}N.\ [\neg \mathtt{enter}(i)]X \wedge \\ & \quad [\mathtt{enter}(i)]\nu Y.\ ([\exists j{:}N.\ \mathtt{enter}(j)]\bot \wedge [\mathtt{idle}(i)]X \wedge [\neg \mathtt{idle}(i)]Y)) \end{aligned} \tag{ϕ}$$

The equation system encoding the model checking problem $TR(l, t) \models \phi$ is, after some minor logical rewriting, as follows:

$$\begin{aligned} & (\nu X(l{:}\mathcal{L}(\{0,1,2\}), t{:}N) = \\ & \quad \forall i{:}N.\ ((t = i \wedge l.t = 1) \implies Y(l[2/t], t, t)) \wedge ((l.t = 2 \implies X(l[0/t], t))) \\ & \quad \wedge((l.t = 1 \wedge i \neq t) \implies X(l[2/t], t)) \wedge (l.t = 0 \implies X(l[1/t], \mathsf{next}(t))) \\ & \quad \wedge(l.t = 0 \implies X(l, \mathsf{next}(t)))) \\ & (\nu Y(l{:}\mathcal{L}(\{0,1,2\}), t, i{:}N) = \\ & \quad ((l.t = 1 \wedge \exists j{:}N.\ t = j) \implies \bot) \\ & \quad \wedge((t = i \wedge l.t = 2) \implies X(l[0/t], t)) \wedge ((t \neq i \wedge l.t = 2) \implies Y(l[0/t], t, i)) \\ & \quad \wedge(l.t = 1 \implies Y(l[2/t], t, i)) \wedge (l.t = 0 \implies Y(l[1/t], \mathsf{next}(t), i)) \\ & \quad \wedge(l.t = 0 \implies Y(l, \mathsf{next}(t), i))) \end{aligned}$$

Here too, symbolic approximation does not converge, not least because the equations for X and Y are mutually dependent. Global invariants are required as both equations are open. The global invariant f we identify assigns $\top$ to equation X and $l.i = 2 \wedge t = i$ for Y, relating process parameters l, t and the formula parameter i, which intuitively expresses that process i is the process that currently possesses the token and, moreover, is in the critical section. Note that $f(X)$ does not add constraints to X, and, therefore, does also not add constraints to the solution to X.

In the invariant-strengthened equation system, Y can be seen to have the solution $l.t = 2 \wedge t = i \wedge X(l[0/t], t)$. The solution for X can then be found by a substitution of Y into the equation for X; a subsequent symbolic approximation immediately leads to the answer $\top$. Note that Proposition 38 would apply equally well to the resulting equation for X, as $\top$ was identified as the invariant for X.

7.6. Verification of a Cache Coherence Protocol

Following [1], we consider Steve German's Cache Coherence Protocol [25] and one of its essential safety properties, *coherence*. The Cache Coherence Protocol serves $N > 0$ clients, and has a single controlling component. The coherence property says that whenever one client has been granted exclusive access to the cache, no other client may access the cache until the exclusive grant is retracted.

The model we present in Table 2 is taken from [1], but is given directly as an LPE, meaning that all parallelism between the various components has been eliminated in favour of a linear representation; the transformation is elementary. To support readability, we only denote the variables of the LPE that are changed by executing a particular action.

Process P below models the protocol's behaviour when N clients are using the cache. P's parameter *cache* is a list of length N; each element *cache*.n represents a local variable of client n's current access rights: *invalid*, i.e. no access; *shared*, i.e. shared access granted; or *exclusive*, i.e. exclusive access granted. Every client n communicates to the controller via three channels, modelled as a list of size N. Channel c_1 and c_3 are used to send messages of type M_1, respectively M_3 to the controller. Channel c_2, again a list of size N contains a command of type M_2, which the controller last sent to client a n. The remainder of P's variables are local to the controller: k and *cd* are the client and the command currently under processing, e is a Boolean value set to $\top$ when a client has received exclusive access, s is a Boolean list maintaining the clients to which some access has been granted and i is an auxiliary Boolean list used when invalidating grants.

Any client n with no request pending ($c_1.n = \mathit{empty}$), may request shared or exclusive access, if he does not already have it. This is expressed by the first two summands of process P, describing the actions `req_shared` and `req_exclusive`, which write their command in $c_1.n$, for the controller to read. Indeed, the controller reading c_1 and updating its local state (i.e., k

Table 2
The LPE modelling German's Cache Coherence Protocol.

Sort	$M_1 ::= \mathit{empty} \mid \mathit{req\_exclusive} \mid \mathit{req\_shared}$
	$M_2 ::= \mathit{empty} \mid \mathit{invalidate} \mid \mathit{grant\_exclusive} \mid \mathit{grant\_shared}$
	$M_3 ::= \mathit{empty} \mid \mathit{invalidate\_ack}$
	$S ::= \mathit{invalid} \mid \mathit{shared} \mid \mathit{exclusive}$

$$
\begin{aligned}
&P(\mathit{cache}{:}\mathcal{L}(S), c_1{:}\mathcal{L}(M_1), c_2{:}\mathcal{L}(M_2), c_3{:}\mathcal{L}(M_3), k{:}N, e{:}B, i, s{:}\mathcal{L}(B), \mathit{cd}{:}M_1) = \\
&\quad \textstyle\sum n{:}N.\ (n \le N \wedge c_1.n = \mathit{empty} \wedge \mathit{cache}.n = \mathit{invalid}) \\
&\qquad \Longrightarrow \texttt{req\_shared}(n) \cdot P(\mathit{cache} := \mathit{cache}[\mathit{invalid}/n],\ c_1 := c_1[\mathit{req\_shared}/n]) \\
&+ \textstyle\sum n{:}N.\ (n \le N \wedge c_1.n = \mathit{empty} \wedge (\mathit{cache}.n = \mathit{invalid} \vee \mathit{cache}.n = \mathit{shared})) \\
&\qquad \Longrightarrow \texttt{req\_exclusive}(n) \cdot P(c_1 := c_1[\mathit{req\_exclusive}/n]) \\
&+ \textstyle\sum n{:}N.\ (n \le N \wedge c_2.n = \mathit{invalidate} \wedge c_3.n = \mathit{empty}) \\
&\qquad \Longrightarrow \texttt{invalidate\_ack}(n) \cdot \\
&\qquad\quad P(\mathit{cache} := \mathit{cache}[\mathit{invalid}/n],\ c_2 := c_2[\mathit{empty}/n], \\
&\qquad\qquad c_3 := c_3[\mathit{invalidate\_ack}/n]) \\
&+ \textstyle\sum n{:}N.\ (n \le N \wedge c_2.n = \mathit{grant\_shared}) \\
&\qquad \Longrightarrow \texttt{shared}(n) \cdot P(\mathit{cache} := \mathit{cache}[\mathit{shared}/n],\ c_2 := c_2[\mathit{empty}/n]) \\
&+ \textstyle\sum n{:}N.\ (n \le N \wedge c_2.n = \mathit{grant\_exclusive}) \\
&\qquad \Longrightarrow \texttt{exclusive}(n) \cdot P(\mathit{cache} := \mathit{cache}[\mathit{exclusive}/n],\ c_2 := c_2[\mathit{empty}/n]) \\
&+ (\mathit{cd} = \mathit{req\_shared} \wedge \neg e \wedge c_2.k = \mathit{empty}) \\
&\qquad \Longrightarrow \texttt{grant\_shared}(k) \cdot P(c_2 := c_2[\mathit{grant\_shared}/k], \\
&\qquad\quad s := s[\top/k],\ \mathit{cd} := \mathit{empty}) \\
&+ (\mathit{cd} = \mathit{req\_exclusive} \wedge c_2.k = \mathit{empty} \wedge \forall j \le N.\ \neg s.j) \\
&\qquad \Longrightarrow \texttt{grant\_exclusive}(k) \cdot \\
&\qquad\quad P(c_1 := c_2[\mathit{grant\_exclusive}/k],\ e := \top,\ s := s[\top/k],\ \mathit{cd} := \mathit{empty}) \\
&+ \textstyle\sum n{:}N.\ (n \le N \wedge \mathit{cd} = \mathit{empty} \wedge c_1.n \neq \mathit{empty}) \\
&\qquad \Longrightarrow \tau \cdot P(c_1 := c_1[\mathit{empty}/n],\ k := n,\ i := s,\ \mathit{cd} := c_1.n) \\
&+ \textstyle\sum n{:}N.\ (n \le N \wedge i.n \wedge c_2.n = \mathit{empty} \\
&\qquad\quad \wedge (\mathit{cd} = \mathit{req\_exclusive} \vee (\mathit{cd} = \mathit{req\_shared} \wedge e))) \\
&\qquad \Longrightarrow \texttt{invalidate}(n) \cdot P(c_2 := c_2[\mathit{invalidate}/n],\ i := i[\bot/n]) \\
&+ \textstyle\sum n{:}N.\ (n \le N \wedge \mathit{cd} \neq \mathit{empty} \wedge c_3.n = \mathit{invalidate\_ack}) \\
&\qquad \Longrightarrow \tau \cdot P(c_3 := c_3[\mathit{empty}/n],\ k,\ e := \bot,\ s := s[\bot/n])
\end{aligned}
$$

and *cd*) is modelled as an unobservable τ action, the 8th summand of P. Further, the controller will work towards meeting this request, by freeing the cache from the clients currently sharing it (maintained in list s). This is done by first copying s to i (8th summand) and then sending *invalidate* messages to every client left in i (action `invalidate`, 9th summand). An "invalidated" client is removed from i as soon as the *invalidate* command is received and acknowledged by the client (action `invalidate_ack`, 3rd summand) and the acknowledgement received by the controller (action τ, last summand). When the situation eventually allows it (that is, no marked clients left in i), the controller grants the requested access by filling channel 2 of the requester k. This is modelled by actions `grant_shared` and `grant_exclusive` of the 6th and 7th summands. Finally, client k will receive its grant and update its local *cache* variable (action `shared` or `exclusive`, 4th and 5th summands).

We check the following consistency property: whenever a client is granted exclusive access, no other client can gain access until exclusive access is released. This is expressed by the following μ-calculus formula:

$$
\begin{aligned}
&\nu X.\ [\top]X \wedge \forall m{:}N.\ [\texttt{exclusive}(m)](\nu Y.\ [\neg\texttt{invalidate}(m)]Y \\
&\quad \wedge [\exists j{:}N.\ (\texttt{exclusive}(j) \vee \texttt{shared}(j))]\bot)
\end{aligned}
\tag{10}
$$

The above formula is a variation of what is checked in [1,25]. A translation of the above formula into an equation system is given in Table 3.

Due to the increasing complexity of the approximants, it is not feasible to solve this PBES by either manual or automated symbolic approximation. This complexity is mainly due to the operations on lists. We are therefore looking for invariants that would properly formalise our intuitions regarding the links between the many data parameters. Intuitively, equation Y describes the behaviour of the system between the execution of a `grant_exclusive` action and the execution of the next following `invalidate` action. According to the consistency formula (10), the subformula $[\exists j{:}N.\ (\texttt{exclusive}(j) \vee \texttt{shared}(j))]\bot$ should be true in this fragment. Translated to a condition on data instead of actions, this is exactly the last line of Y's equation, $\forall n{:}N.\ (n \le N \implies (c_2.n \neq \mathit{grant\_shared} \wedge c_2.n \neq \mathit{grant\_exclusive}))$. So, we are looking for an invariant that over-approximates this last line and hopefully still characterises all states reachable from an instantiation of Y as occurring in X's equation. Since the intuition of the protocol's behaviour advises that actions `grant_shared`, `shared`, `invalidate_ack` and the last τ should not be enabled in the fragment described by Y, we start by considering the negation of the respective guards as possible invariant. This leads to the following predicate:

$$
\begin{aligned}
(\alpha):\quad & e \wedge s.m \wedge \forall n{:}N.\ (n \le N \implies (c_2.n = \mathit{empty})) \\
& \wedge (\mathit{cd} = \mathit{empty} \vee \forall n{:}N.\ (n \le N \implies (c_3.n = \mathit{empty}))) \\
& \wedge \forall n{:}N.\ (n \le N \wedge n \neq m \implies i.n = \bot)
\end{aligned}
$$

Table 3
Equation system encoding mutual exclusion for the Cache Coherence Protocol.

$$
\begin{array}{l}
\Big(\nu X(cache{:}\mathcal{L}(S), c_1{:}\mathcal{L}(M_1), c_2{:}\mathcal{L}(M_2), c_3{:}\mathcal{L}(M_3), k{:}N, e{:}B, i, s{:}B, cd{:}M_1) = \\
\quad \forall n{:}N.\ (n \leq N \wedge c_1.n = empty \wedge cache.n = invalid) \\
\qquad \Longrightarrow\ X(cache := cache[invalid/n], c_1 := c_1[req\_shared/n]) \\
\wedge \forall n{:}N.\ (n \leq N \wedge c_1.n = empty \wedge (cache.n = invalid \vee cache.n = shared)) \\
\qquad \Longrightarrow\ X(c_1 := c_1[req\_exclusive/n]) \\
\wedge \forall n{:}N.\ (n \leq N \wedge c_2.n = invalidate \wedge c_3.n = empty) \\
\qquad \Longrightarrow\ X(cache := cache[invalid/n], c_2 := c_2[empty/n], \\
\qquad\qquad c_3 := c_3[invalidate\_ack/n]) \\
\wedge \forall n{:}N.\ (n \leq N \wedge c_2.n = grant\_shared) \\
\qquad \Longrightarrow\ X(cache := cache[shared/n], c_2 := c_2[empty/n]) \\
\wedge \forall n{:}N.\ (n \leq N \wedge c_2.n = grant\_exclusive) \\
\qquad \Longrightarrow\ X(cache := cache[exclusive/n], c_2 := c_2[empty/n]) \\
\wedge (n \leq N \wedge cd = req\_shared \wedge \neg e \wedge c_2.k = empty) \\
\qquad \Longrightarrow\ X(c_2 := c_2[grant\_shared/k], s := s[\top/k], cd := empty) \\
\wedge (cd = req\_exclusive \wedge c_2.k = empty \wedge \forall j \leq N.\ \neg s.j) \\
\qquad \Longrightarrow\ X(c_2 := c_2[grant\_exclusive/k], e := \top, s := s[\top/k], cd := empty) \\
\wedge \forall n{:}N.\ (n \leq N \wedge cd = empty \wedge c_1.n \neq empty) \\
\qquad \Longrightarrow\ X(c_1 := c_1[empty/n], k := n, i := s, cd := c_1.n) \\
\wedge \forall n{:}N.\ (n \leq N \wedge i.n \wedge c_2.n = empty \\
\qquad\quad \wedge (cd = req\_exclusive \vee (cd = req\_shared \wedge e))) \\
\qquad \Longrightarrow\ X(c_2 := c_2[invalidate/n], i := i[\bot/n]) \\
\wedge \forall n{:}N.\ (n \leq N \wedge cd \neq empty \wedge c_3.n = invalidate\_ack) \\
\qquad \Longrightarrow\ X(c_3 := c_3[empty/n], k, e := \bot, s := s[\bot/n]) \\
\wedge \forall n{:}N.(n \leq N \wedge c_2.n = grant\_exclusive) \\
\qquad \Longrightarrow\ Y(cache[exclusive/n], c_1, c_2[empty/n], c_3, k, e, i, s, cd, n)\Big)
\end{array}
$$

$$
\begin{array}{l}
\Big(\nu Y(cache{:}\mathcal{L}(S), c_1{:}\mathcal{L}(M_1), c_2{:}\mathcal{L}(M_2), c_3{:}\mathcal{L}(M_3), k{:}N, e{:}B, i, s{:}B, cd{:}M_1, m{:}N) = \\
\quad \forall n{:}N.\ (n \leq N \wedge c_1.n = empty \wedge cache.n = invalid) \\
\qquad \Longrightarrow\ Y(cache := cache[invalid/n], c_1 := c_1[req\_shared/n]) \\
\wedge \forall n{:}N.\ (n \leq N \wedge c_1.n = empty \wedge (cache.n = invalid \vee cache.n = shared)) \\
\qquad \Longrightarrow\ Y(c_1 := c_1[req\_exclusive/n]) \\
\wedge \forall n{:}N.\ (n \leq N \wedge c_2.n = invalidate \wedge c_3.n = empty) \\
\qquad \Longrightarrow\ Y(cache := cache[invalid/n], c_2 := c_2[empty/n], \\
\qquad\qquad c_3 := c_3[invalidate\_ack/n]) \\
\wedge \forall n{:}N.\ (n \leq N \wedge c_2.n = grant\_shared) \\
\qquad \Longrightarrow\ Y(cache := cache[shared/n], c_2 := c_2[empty/n]) \\
\wedge \forall n{:}N.\ (n \leq N \wedge c_2.n = grant\_exclusive) \\
\qquad \Longrightarrow\ Y(cache := cache[exclusive/n], c_2 := c_2[empty/n]) \\
\wedge (n \leq N \wedge cd = req\_shared \wedge \neg e \wedge c_2.k = empty) \\
\qquad \Longrightarrow\ Y(c_2 := c_2[grant\_shared/k], s := s[\top/k], cd := empty) \\
\wedge (cd = req\_exclusive \wedge c_2.k = empty \wedge \forall j \leq N.\ \neg s.j) \\
\qquad \Longrightarrow\ Y(c_2 := c_2[grant\_exclusive/k], e := true, \\
\qquad\qquad s := s[\top/k], cd := empty) \\
\wedge \forall n{:}N.\ (n \leq N \wedge cd = empty \wedge c_1.n \neq empty) \\
\qquad \Longrightarrow\ Y(c_1 := c_1[empty/n], k := n, i := s, cd := c_1.n) \\
\wedge \forall n{:}N.\ (n \leq N \wedge n \neq m \wedge i.n \wedge c_2.n = empty \\
\quad\ \wedge (cd = req\_exclusive \vee (cd = req\_shared \wedge e))) \\
\qquad\ \Longrightarrow\ Y(c_2 := c_2[invalidate/n], i := i[\bot/n]) \\
\wedge \forall n{:}N.\ (n \leq N \wedge cd \neq empty \wedge c_3.n = invalidate\_ack) \\
\qquad \Longrightarrow\ Y(c_3 := c_3[empty/n], e := \bot, s := s[\bot/n]) \\
\wedge \forall n{:}N.\ (n \leq N\ \Longrightarrow\ (c_2.n \neq grant\_shared \wedge c_2.n \neq grant\_exclusive))\Big)
\end{array}
$$

α satisfies the sufficient condition of Property 29 for the equation of Y and is therefore a local invariant for Y. Since α implies $\forall n{:}N.\ (n \leq N \implies (c_2.n \neq grant\_shared \wedge c_2.n \neq grant\_exclusive))$, the equation of Y after strengthening with α has the same form as before strengthening, except the last line changes into α. By applying Proposition 38, we obtain that the solution of the strengthened Y equation must be α.

We now need to find a predicate β such that f defined as $(f(X) = \beta, f(Y) = \alpha)$ would be a global invariant for our PBES. Ideally, β would again allow the application of Proposition 38, so we start with filling in the solution for Y in X and find that the last line of X's equation becomes after an immediate logical rewriting,

$$
\begin{array}{l}
(\lambda)\ :\ \forall n{:}N.(n \leq N \wedge c_2.n = grant\_exclusive) \\
\qquad \Longrightarrow\ e \wedge s.n \wedge \forall j{:}N.\ (j \leq N \wedge j \neq n\ \Longrightarrow\ (c_2.j = empty \wedge i.j = \bot)) \\
\qquad\quad \wedge (cd = empty \vee \forall j{:}N.\ (j \leq N\ \Longrightarrow\ (c_3.j = empty)))
\end{array}
$$

Unfortunately, λ is not an invariant for X, therefore we look for a stronger formula that would satisfy the conditions of an invariant without restricting too much the characterised state space. In [1], the following process invariants are used:

$$
\begin{array}{ll}
(\kappa 1) & e \implies |\{n|s.n\}| \leq 1 \\
(\kappa 2) & \forall n{:}N.\ n \leq N \implies \neg(i.n \wedge c_2.n = invalidate) \\
(\kappa 3) & \forall n{:}N.\ n \leq N \implies \neg(i.n \wedge c_3.n \neq empty) \\
(\kappa 4) & \forall n{:}N.\ n \leq N \implies \neg(c_2.n \neq empty \wedge c_3.n \neq empty) \\
(\kappa 5) & \forall n{:}N.\ n \leq N \implies (\neg s.n \implies (\neg i.n \wedge c_2.n = empty \\
 & \qquad\qquad \wedge c_3.n = empty \wedge cache.n = invalid)) \\
(\kappa 6) & ((cd \neq req\_exclusive \wedge (\neg e \vee cd \neq req\_shared)) \\
 & \qquad \vee \forall n{:}N.\ n \leq N \implies \neg s.n) \\
 & \implies (\forall n{:}N.\ n \leq N \implies (c_2.n \neq invalidate \wedge c_3.n = empty)) \\
(\kappa 7) & e \implies |\{n|i.n\}| \leq 1 \\
(\kappa 8) & \forall n{:}N.\ n \leq N \implies \neg(c_3.n = invalidate\_ack \wedge cache.n \neq invalid))
\end{array}
$$

Indeed, as claimed in [1], $\kappa = \bigwedge_{i:1\leq i\leq 8} \kappa_i$ is a process invariant for process P specified in Table 2, and therefore, following Theorem 51, it is also a local invariant for both X and Y. Let us take as β the predicate $\kappa \wedge (\exists n{:}N.\ (n \leq N \implies c_2.n = grant\_exclusive) \implies e)$. By verifying the assumptions of Property 29 for both X and Y, we can check that (β, α) is a global invariant for the initial PBES. The only not obvious point in this task is checking that the formula $(\exists n{:}N.\ (n \leq N \implies c_2.n = grant\_exclusive) \implies e)$ is preserved by the summand of X's equation where the guard is $cd \neq empty \wedge c_3.n = invalidate\_ack$ and, in the right-hand side parameter list, $e = \bot$. We reason as follows: Suppose $\exists j{:}N.\ (j \leq N \implies c_2.j = grant\_exclusive)$ and e are true beforehand. Then it follows from $(\kappa 5)$ that $s.j = \top$. Note also that $c_3.n = invalidate\_ack$ implies (due to invariant $\kappa 5$) $s.n = \top$. So, from $e = \top$ and $(\kappa 1)$ follows that $j = n$, meaning that $c_2.n = grant\_exclusive$, which, via $(\kappa 4)$, implies $c_3.n = empty$, contradicting the guard $c_3.n = invalidate\_ack$. Therefore, $e = \bot$ must hold already in the guard, and since c_2 is not modified, the implication $(\exists n{:}N.\ (n \leq N \implies c_2.n = grant\_exclusive) \implies e)$ does not change its truth value.

We proceed to solve the PBES strengthened with (β, α). It turns out that α is a solution for the new equation of Y and, consequently, after substituting α in the (β-strengthened) equation of X, also that β is a solution for X. For both equations, we made use of Proposition 38.

Finally, instantiating β with parameters corresponding to the initial state of the protocol yields:

$$
\begin{array}{l}
\beta(invalid \ldots invalid, empty \ldots empty, empty \ldots empty, empty \ldots empty, \\
\qquad 0, \bot, \bot \ldots \bot, \bot \ldots \bot, empty) = \top
\end{array}
$$

Note the modular approach allowed by the PBES description. We could first find an invariant and solution for a small subsystem (namely, the subsystem triggered by action `grant_exclusive`(m) and ended by action `invalidate`(m)), for some process m. This invariant could then be used to build a global invariant and find the solution of the whole PBES.

However, checking the correctness of invariants on such large specifications is challenging and error-prone and would definitely benefit from tool support.

8. Conclusion

Techniques and concepts for solving PBESs have been studied in detail [15]. Among these is the concept of *invariance*, which has been instrumental in solving verification problems that were studied in e.g. [15,4]. In this paper, we further studied the notion of invariance and show that the accompanying theory is inappropriate for PBESs in which *open* equations occur. We have proposed a stronger notion of invariance, called *global invariance*, and phrased an associated invariance theorem. We moreover have shown that our notion of invariance is preserved by three important solution-preserving PBES manipulations. This means that, unlike the notion of invariance of [15], global invariants can be used in combination with these manipulations when solving equation systems. As a side-result, we obtain a partial answer to an open question, concerning a specific pattern for PBESs, first put forward in [15].

We continued by demonstrating that invariants for processes automatically yield global invariants in the PBESs resulting from two standard verification encodings, viz. the encoding of the first-order modal μ-calculus model checking problem and the encoding of various process equivalences for two (possibly infinite) transition systems. This means that in the PBES verification methodology, one can take advantage of established techniques for checking and discovering process invariants.

We suspect that many techniques for discovering process invariants, e.g., [24,25], can be put to use for (automatically) discovering global invariants in PBESs. Of course, transferring such techniques may not be straightforward and requires additional research. We take steps in this direction in [23], in which we identify a class of invariants that can be derived automatically. This class of invariants consists of conjunctions of simple formulae of the form $d = v$, where d is a parameter and v is a constant. We are currently investigating algorithms for a larger class of invariants. Closely related to the quest of finding techniques for discovering invariants, is the search to find an alternative invariance condition that does not require a quantification over arbitrary predicate variable environments. Steps towards this goal are taken in [23], in which we generalise Property 29 by identifying a condition that is more liberal, yet still stronger than our invariance condition. We believe that, under mild restrictions, this more liberal condition coincides with our invariance condition. Identifying these restrictions is another line of research we are currently pursuing.

Acknowledgements

The authors would like to thank Jan Friso Groote for providing a counterexample (see Example 14) to a conjecture involving syntactic substitution. This research has been partially funded by the Netherlands Organisation for Scientific Research (NWO) under FOCUS/BRICKS grant number 642.000.602.

References

[1] K. Baukus, Y. Lakhnech, K. Stahl, Parameterized verification of a cache coherence protocol: Safety and liveness, in: VMCAI, 2002, pp. 317–330.
[2] M.A. Bezem, J.F. Groote, Invariants in process algebra with data, in: B. Jonsson, J. Parrow (Eds.), Proceedings Concur'94, in: LNCS, vol. 836, Springer Verlag, 1994, pp. 401–416.
[3] A.R. Bradley, Z. Manna, H.B. Sipma, The polyranking principle, in: ICALP,2005, pp. 1349–1361.
[4] T. Chen, B. Ploeger, J. van de Pol, T.A.C. Willemse, Equivalence checking for infinite systems using parameterized boolean equation systems, in: Proc. 18th Int'l Conference on CONCUR, in: LNCS, vol. 4703, Springer, 2007, pp. 120–135.
[5] A. van Dam, B. Ploeger, T.A.C. Willemse, Instantiation for parameterised boolean equation systems, in: J.S. Fitzgerald, A.E. Haxthausen, H. Yenigün (Eds.), Theoretical Aspects of Computing – ICTAC 2008 (5th International Colloquium, Istanbul, Turkey), in: Lecture Notes in Computer Science, vol. 5160, Springer-Verlag, 2008, pp. 440–454.
[6] Y. Fang, N. Piterman, A. Pnueli, L.D. Zuck, Liveness with invisible ranking, International Journal on Software Tools for Technology Transfer (STTT) 8 (3) (2006) 261–279.
[7] R.W. Floyd, Assigning meanings to programs, in: Mathematical Aspects of Computer Science, in: Proceedings of Symposia in Applied Mathematics, vol. 19, American Mathematical Society, 1967, pp. 19–32.
[8] A. Fujioka, T. Okamoto, K. Ohta, A practical secret voting scheme for large scale elections, in: Proc. 3rd Workshop on the Theory and Application of Cryptographic Techniques, in: LNCS, vol. 718, 1992.
[9] M.M. Gallardo, C. Joubert, P. Merino, Implementing influence analysis using parameterised boolean equation systems, in: Proceedings of ISOLA'06, November, IEEE Computer Society Press, 2006.
[10] H. Garavel, R. Mateescu, F. Lang, W. Serwe, CADP 2006: A toolbox for the construction and analysis of distributed processes, in: W. Damm, H. Hermanns (Eds.), Proceedings of CAV, in: LNCS, vol. 4590, Springer, 2007, pp. 158–163.
[11] J.F. Groote, R. Mateescu, Verification of temporal properties of processes in a setting with data, in: Proc. of AMAST, in: LNCS, vol. 1548, Springer, 1999, pp. 74–90.
[12] J.F. Groote, A.H.J. Mathijssen, M.A. Reniers, Y.S. Usenko, M.J. van Weerdenburg, The formal specification language mCRL2, in: E. Brinksma, D. Harel, A. Mader, P. Stevens, R. Wieringa (Eds.), Methods for Modelling Software Systems, MMOSS, in: Dagstuhl Seminar Proceedings, vol. 06351, 2007.
[13] J.F. Groote, A. Ponse, The syntax and semantics of mCRL, in: A. Ponse, C. Verhoef, and S.F.M. van Vlijmen, editors, Algebra of Communicating Processes, Workshops in Computing, 1994, pp. 26–62.
[14] J.F. Groote, T.A.C. Willemse, Model-checking processes with data, Sci. Comput. Program 56 (3) (2005) 251–273.
[15] J.F. Groote, T.A.C. Willemse, Parameterised boolean equation systems, Theor. Comput. Sci 343 (3) (2005) 332–369.
[16] C.A.R. Hoare, An axiomatic basis for computer programming, Communications of the ACM 12 (10) (1969) 576–585.
[17] D. Knuth, The Art of Computer Programming, in: Sorting and Searching chapter 5, section 2.1: Sorting by Insertion, vol. 3, Addison-Wesley, 1998, pp. 80–105.
[18] D. Kozen, Results on the propositional mu-calculus, Theoret. Comput. Sci. 27 (1983) 333–354.
[19] A. Mader, Verification of Modal Properties Using Boolean Equation Systems. Ph.D. thesis, Technische Universität München, 1997.
[20] R. Mateescu, Local model-checking of an alternation-free value-based modal mu-calculus, in: Proc. 2nd Int'l Workshop on VMCAI, September 1998.
[21] R. Mateescu, Vérification des propriétés temporelles des programmes paralléles, Ph.D. thesis, Institut National Polytechnique de Grenoble, 1998.
[22] S. Orzan, T.A.C. Willemse, Invariants for parameterised boolean equation systems (extended abstract), in: F. van Breugel, M. Checkik (Eds.), Proceedings of CONCUR'08, in: LNCS, vol. 5201, Springer-Verlag, 2008, pp. 187–202.
[23] S.M. Orzan, J.W. Wesselink, T.A.C. Willemse, Static analysis techniques for parameterised boolean equation systems, in: S. Kowalewski, A. Philippou (Eds.), Proceedings of TACAS, in: LNCS, vol. 5505, 2009, pp. 230–245.
[24] S. Pandav, K. Slind, G. Gopalakrishnan, Counterexample guided invariant discovery for parameterized cache coherence verification, in: Proceedings CHARME, in: LNCS, vol. 3725, Springer, 2005, pp. 317–331.
[25] A. Pnueli, S. Ruah, L. Zuck, Automatic deductive verification with invisible invariants, in: T. Margaria, W. Yi (Eds.), Proceedings TACAS, in: LNCS, vol. 2031, 2001, pp. 82–97.
[26] S. Sankaranarayanan, H.B. Sipma, Z. Manna, Constructing invariants for hybrid systems, Form. Methods Syst. Des. 32 (1) (2008) 25–55.
[27] A. Tarski, A lattice-theoretical fixpoint theorem and its applications, Pacific J. Math. 5 (2) (1955) 285–309.
[28] J.C. van de Pol, 2007, Personal communication.
[29] D. Zhang, R. Cleaveland, Fast generic model-checking for data-based systems, in: F. Wang (Ed.), FORTE, in: LNCS, vol. 3731, Springer, 2005, pp. 83–97.