Invariants for Parameterised Boolean Equation Systems

Simona Orzan and Tim A.C. Willemse

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

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. Several examples are provided that illustrate the advantages of using global invariants in various verification problems.

1 Introduction

Parameterised Boolean Equation Systems (PBESs), introduced in [20, 19], 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 [26]; 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) [18], 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 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 such 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 weak 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. We remedy this situation by introducing the concept of a *global invariant*, and 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 *Cache Coherence Protocol* from the literature [1, 22]. The examples vary in complexity, and illustrate various verification problems. Many examples involve parametric systems, meaning that the verifications are conducted over *all instances* of these systems.

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 fullfill 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, 21]; 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.

It turns out that invariants provide the foundation of many mature verification methodologies aiming to tackle complex cases, such as networks of parameterised systems [21, 22, 6], various types of equivalence checks between reactive system [2] and for infinite data domains in general, such as hybrid systems [23]. 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 are universally applicable to the problems that can be encoded in them.

Several works, like [21, 6, 23] focus on the automated and even automatic discovery of invariants for specialised classes of specifications and properties. It is likely that many of these techniques can be ported to work for specific PBESs as well. 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

Predicate formulae are basically ordinary predicates extended with predicate variables.

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

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

where b is a data term of Boolean sort $\mathbb{B}$, 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 $\boldsymbol{d_X}$ of sort $\boldsymbol{D_X}$; $\boldsymbol{e}$ is a vector of data terms of the sort $\boldsymbol{D_X}$. The data variables occurring in a predicate formula are taken from a set $\mathcal{D}$ of data variables.

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 $\mathsf{occ}(\phi)$.

Remark 1. 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 $\mathbb{B}$.

Remark 2. As usual, we use predicate variables X to which we associate a single variable d_X of sort D_X instead of vectors $\boldsymbol{d_X}$ of sort $\boldsymbol{D_X}$ 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 2. *Let θ be a* predicate environment *assigning a function of type $D_X \to \mathbb{B}$ 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:*

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

Remark 3. 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 3. *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:

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

Lemma 1. *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:

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

Example 1. 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 1. 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 2. *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:

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

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

$$\begin{array}{ll} & [\![\psi]\!]\theta\varepsilon = [\![\rho]\!]\theta\varepsilon \\ \Rightarrow & \{\text{Lemma 1}\} \\ & \theta[[\![\psi_{\langle d_X \rangle}]\!]\theta\varepsilon/X] = \theta[[\![\rho_{\langle d_X \rangle}]\!]\theta\varepsilon/X] \\ \Rightarrow & \\ & [\![\chi]\!]\theta[[\![\psi_{\langle d_X \rangle}]\!]\theta\varepsilon/X] = [\![\chi]\!]\theta[[\![\rho_{\langle d_X \rangle}]\!]\theta\varepsilon/X] \\ \Leftrightarrow & \{\text{Property 1}\} \\ & [\![\chi[\psi_{\langle d_X \rangle}/X]]\!]\theta\varepsilon = [\![\chi[\rho_{\langle d_X \rangle}/X]]\!]\theta\varepsilon \end{array}$$

□

Lemma 3. *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:

$$\begin{array}{ll} & [\![\psi[\chi_{\langle d_X \rangle}/X]]\!]\theta\varepsilon \\ = & \{\text{property 1}\} \\ & [\![\psi]\!]\theta[[\![\chi_{\langle d_X \rangle}]\!]\theta\varepsilon/X]\varepsilon \\ = & \{\psi \leftrightarrow \rho, \text{ so } \psi \text{ and } \rho \text{ are indistinguishable for all predicate environments }\} \\ & [\![\rho]\!]\theta[[\![\chi_{\langle d_X \rangle}]\!]\theta\varepsilon/X]\varepsilon \\ = & \{\text{Property 1}\} \\ & [\![\rho[\chi_{\langle d_X \rangle}/X]]\!]\eta\varepsilon \end{array}$$

□

Lemma 4. *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 1 once:

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

□

Lemma 5. *Let ϕ,ψ,ρ be arbitrary predicate formulae. Whenever $X \notin \mathsf{occ}(\rho)$ 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 $X \notin \mathsf{occ}(\rho)$ 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 below derivation utilises Property 1 exactly once.

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

□

The interplay between the equivalence $\leftrightarrow$ on predicates and the notion of syntactic substitutions is quite delicate. For instance, one may believe that when $\phi[\rho_{\langle d_X\rangle}/X] \leftrightarrow \phi[\psi_{\langle d_X\rangle}/X]$, then $\phi[(\rho \wedge X(d_X))_{\langle d_X\rangle}/X] \leftrightarrow \phi[(\psi \wedge X(d_X))_{\langle d_X\rangle}/X]$ also holds. However, the following example shows that this consequence is invalid:

Example 2. 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 5. 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 substitions of the form $\phi[\psi_{1\langle d_{X_1}\rangle}/X_1][\psi_{2\langle d_{X_2}\rangle}/X_2]\dots[\phi_{n\langle d_{X_n}\rangle}/X_n]$, where all predicate formulae ψ_i are similar, as follows:

Definition 4. *Let $V = \langle X_1, \ldots, X_n \rangle$ be a vector of predicate variables and let ϕ_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:*

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

In case we have that for all ϕ_i, only variable X_i occurs in ϕ_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 substitition. This is expressed by the following lemma.

Lemma 6. *Let $X_1, \ldots, X_n$ be distinct predicate variables, and let ϕ_i, for $1 \leq i \leq n$, 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 5. *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:*

$$\begin{array}{ll} [\![\epsilon]\!]\theta\varepsilon & =_{def} \theta \\ [\![(\sigma X(d_X{:}D_X) = \phi)\mathcal{E}]\!]\theta\varepsilon & =_{def} [\![\mathcal{E}]\!](\theta\Big[\sigma\mathcal{X}{\in}[D_X \to \mathbb{B}].\ [\![\phi_{\langle d_X \rangle}]\!]([\![\mathcal{E}]\!]\theta[\mathcal{X}/X])\varepsilon/X\Big])\varepsilon \end{array}$$

Note that the fixed points are taken over the complete lattice of functions $([D_X \to \mathbb{B}], \sqsubseteq)$ 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

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

is a monotone operator [14, 15, 11]. The existence of the (extremal) fixed points of this operator in the lattice $([D_X \to \mathbb{B}], \sqsubseteq)$ follows immediately from Tarski's fixed point Theorem [24]. A standard, constructive technique for computing a fixed point is by means of a transfinite approximation over the ordinals (see, e.g. [18]).

Definition 6. *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:*

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

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, 25]:

Lemma 7. *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*

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

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

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

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

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

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, 25].

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 [18].

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 1 (See [15]). *Let ϕ be a predicate formula, $k{:}\mathbb{N}$ 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 7. *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 2 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 1 (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 $X \notin \mathsf{occ}(\chi)$:*

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

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:

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

□

Theorem 1 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 [19, 14]. A second issue is that invariants may "break" as a result of a substitution:

Example 3. Consider the following (constructed) closed equation system:

$$\begin{array}{l} (\mu X(n{:}\mathbb{N}) = n \geq 2 \wedge Y(n)) \\ (\mu Y(n{:}\mathbb{N}) = Z(n) \vee Y(n+1)) \\ (\mu Z(n{:}\mathbb{N}) = n < 2 \vee Y(n-1)) \end{array} \qquad (1)$$

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):

$$\begin{array}{l} (\mu X(n{:}\mathbb{N}) = n \geq 2 \wedge Y(n)) \\ (\mu Y(n{:}\mathbb{N}) = n < 2 \vee Y(n-1) \vee Y(n+1)) \\ (\mu Z(n{:}\mathbb{N}) = n < 2 \vee Y(n-1)) \end{array} \qquad (2)$$

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):

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

Using this solution for Y in the equation for X in (1), and solving the resulting equation system leads to the solution $\lambda v{\in}\mathbb{N}.\ v \geq 2$ for X and $\lambda v{\in}\mathbb{N}.\ \top$ for Y and Z. A weakness of Theorem 1 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 1. 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 $\lambda v{\in}\mathbb{N}.\ \bot$ for X. Theorem 40 of [15] is ungainly as it even introduces extra equations. □

Example 3 shows that identified invariants (cf. [15]) fail to remain invariants when substitution is exercised on the equation system, and, more importantly, that Theorem 1 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 Def. 7 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 8. *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 2. *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:

$$
\begin{array}{ll}
& (f(X) \wedge \phi) \left[_{X_i \in W \cup \{X_j\}} (f(X_i) \wedge X_i(d_{X_i}))_{\langle d_{X_i} \rangle} / X_i\right] \\
\leftrightarrow & \{\text{Property of consecutive substitution }\} \\
& ((f(X) \wedge \phi) \left[_{X_i \in W} (f(X_i) \wedge X_i(d_{X_i}))_{\langle d_{X_i} \rangle} / X_i\right]) \\
& \qquad [(f(X_j) \wedge X_j(d_{X_j}))_{\langle d_{X_j} \rangle} / X_j] \\
\leftrightarrow & \{\text{Lemma 3 and (IH)}\} \\
& ((f(X) \wedge \phi)) \, [(f(X_j) \wedge X_j(d_{X_j}))_{\langle d_{X_j} \rangle} / X_j] \\
\leftrightarrow & \{\text{Lemma 3 and } f \text{ is a global invariant }\} \\
& ((f(X) \wedge \phi) \left[_{X_i \in V} (f(X_i) \wedge X_i(d_{X_i}))_{\langle d_{X_i} \rangle} / X_i\right]) \\
& \qquad [(f(X_j) \wedge X_j(d_{X_j}))_{\langle d_{X_j} \rangle} / X_j] \\
\leftrightarrow & \{\text{Property of consecutive substitution}\} \\
& ((f(X) \wedge \phi) \left[_{X_i \in V \setminus \{X_j\}} (f(X_i) \wedge X_i(d_{X_i}))_{\langle d_{X_i} \rangle} / X_i\right] \\
& \qquad [(f(X_j) \wedge X_j(d_{X_j}))_{\langle d_{X_j} \rangle} / X_j]) \, [(f(X_j) \wedge X_j(d_{X_j}))_{\langle d_{X_j} \rangle} / X_j] \\
\leftrightarrow & \{\text{Lemma 4}\} \\
& ((f(X) \wedge \phi) \left[_{X_i \in V \setminus \{X_j\}} (f(X_i) \wedge X_i(d_{X_i}))_{\langle d_{X_i} \rangle} / X_i\right] \\
& \qquad [(f(X_j) \wedge f(X_j) \wedge X_j(d_{X_j}))_{\langle d_{X_j} \rangle} / X_j]) \\
\leftrightarrow & \{\text{idempotence of } \wedge\} \\
& ((f(X) \wedge \phi) \left[_{X_i \in V} (f(X_i) \wedge X_i(d_{X_i}))_{\langle d_{X_i} \rangle} / X_i\right] \\
& \qquad [(f(X_j) \wedge X_j(d_{X_j}))_{\langle d_{X_j} \rangle} / X_j]) \\
\leftrightarrow & \{\text{Property of consecutive substition; } f \text{ is a global invariant}\} \\
& f(X) \wedge \phi
\end{array}
$$

□

Corollary 1. *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 4. 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: $(\nu X(n{:}\mathbb{N}) = Y(n-1))(\mu Y(n{:}\mathbb{N}) = X(n+1))$. 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* occuring in a formula ϕ:

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

Property 2. 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 substition, 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:

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

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

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

- 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 2 is not a necessary condition. For instance, the equation system given by the single (trivial) equation $(\mu X(n{:}\mathbb{N}) = X(n{+}1) \vee \top)$ does not fulfil the condition of Property 2. 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 8).

Property 3. 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 $f{:}V \rightarrow \mathsf{Pred}$ be a simple function such that, for all i, $f(X) \wedge \chi \wedge \psi_i \rightarrow f(X_i)[e_i/d_{X_i}]$. 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:

$$
\begin{aligned}
& (f(X) \wedge \phi) \left[_{X_i \in V} (f(X_i) \wedge X_i(d_{X_i}))_{\langle d_{X_i} \rangle} / X_i\right] \\
\leftrightarrow\ & \{\text{expansion of } \phi,\ f(X) \text{ and } \chi \text{ are simple}\} \\
& f(X) \wedge \chi \wedge \textstyle\bigwedge_{i \in I} (\psi_i \implies f(X_i)[e_i/d_{X_i}] \wedge X_i(e_i)) \\
\leftrightarrow\ & \{\text{for any } \alpha, \beta, \gamma\text{: } (\alpha \implies \beta \wedge \gamma) \leftrightarrow (\alpha \implies \beta \wedge \alpha \implies \gamma)\} \\
& f(X) \wedge \chi \wedge \textstyle\bigwedge_{i \in I} (\psi_i \implies f(X_i)[e_i/d_{X_i}]) \wedge \textstyle\bigwedge_{i \in I} (\psi_i \implies X_i(e_i)) \\
\leftrightarrow\ & \{\text{for any } \alpha, \beta, \gamma\text{: } (\alpha \wedge (\beta \implies \gamma)) \leftrightarrow \alpha \wedge ((\alpha \wedge \beta) \implies \gamma)\} \\
& f(X) \wedge \chi \wedge \textstyle\bigwedge_{i \in I} ((f(X) \wedge \chi \wedge \psi_i) \implies f(X_i)[e_i/d_{X_i}]) \wedge \textstyle\bigwedge_{i \in I} (\psi_i \implies X_i(e_i)) \\
\leftrightarrow\ & \{\text{for all } i,\ f(X) \wedge \chi \wedge \psi_i \rightarrow f(X_i)[e_i/d_{X_i}]\} \\
& f(X) \wedge \chi \wedge \top \wedge \textstyle\bigwedge_{i \in I} (\psi_i \implies X_i(e_i)) \\
& \{\text{definition of } \phi\} \\
\leftrightarrow\ & f(X) \wedge \phi.
\end{aligned}
$$

□

Invariants can be combined using logical connectives $\wedge$ and $\vee$. Let $f, g{:}V \rightarrow \mathsf{Pred}$ 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 8. *Let ϕ be an arbitrary predicate formula and let $f, g{:}V \rightarrow \mathsf{Pred}$ 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:

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

Proof. Let $f, g{:}V \rightarrow \mathsf{Pred}$ 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:

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

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.

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

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 4. 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 8. □

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 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 strenghtening 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 9. *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. $\mathsf{occ}(\phi) \subseteq V$
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:

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

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

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

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 1: 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 1: 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:

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

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 10. *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* $\eta_1, \eta_2, \varepsilon$*:*

$$\forall Y \in V:\ [\![(f(Y) \wedge Y(d_Y))]\!]\eta_1\varepsilon = [\![(f(Y) \wedge Y(d_Y))]\!]\eta_2\varepsilon$$

implies

$$[\![(f(X) \wedge \phi)]\!]\eta_1\varepsilon = [\![(f(X) \wedge \phi)]\!]\eta_2\varepsilon$$

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

$$\begin{array}{l}
[\![f(X) \wedge \phi]\!]\eta_1\varepsilon \\
= \{\text{Assumption on } f\text{; definition of } \leftrightarrow\} \\
[\![(f(X) \wedge \phi)\ \left[_{X_i \in V}(f(X_i) \wedge X_i(d_{X_i}))_{\langle d_{X_i} \rangle}/X_i\right]\ ]\!]\eta_1\varepsilon \\
= \{\text{Property 1 for every } X_i \in V\ \} \\
[\![f(X) \wedge \phi]\!]\eta_1[[\![(f(X_1) \wedge X_1(d_{X_1}))_{\langle d_{X_1} \rangle}]\!]\eta_1\varepsilon/X_1]\dots \\
\qquad [[\![(f(X_n) \wedge X_n(d_{X_n}))_{\langle d_{X_n} \rangle}]\!]\eta_1\varepsilon/X_n] \\
= \{\text{Assumption on } \eta_1, \eta_2\} \\
[\![f(X) \wedge \phi]\!]\eta_2[[\![(f(X_1) \wedge X_1(d_{X_1}))_{\langle d_{X_1} \rangle}]\!]\eta_2\varepsilon/X_1]\dots \\
\qquad [[\![(f(X_n) \wedge X_n(d_{X_n}))_{\langle d_{X_n} \rangle}]\!]\eta_2\varepsilon/X_n] \\
= \{\text{Property 1 for every } X_i \in V\ \} \\
[\![(f(X) \wedge \phi)\ \left[_{X_i \in V}(f(X_i) \wedge X_i(d_{X_i}))_{\langle d_{X_i} \rangle}/X_i\right]\ ]\!]\eta_2\varepsilon \\
= \{\text{Assumption on } f\text{; definition of } \leftrightarrow\} \\
[\![f(X) \wedge \phi]\!]\eta_2\varepsilon
\end{array}$$

□

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

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

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 2. *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 $X \in V$*:*

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

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 $Z \in V$:

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

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

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

   Likewise, we derive:

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

   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:

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

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

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

   Using Lemma 9 and our assumptions, we find:

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

Using Lemma 10, our assumptions and the induction hypothesis, we find:

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

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

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

Corollary 2. *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

$$[\![\phi]\!](\,[\![\mathcal{E}]\!]\eta\varepsilon)\varepsilon = [\![\phi]\!](\,[\![\mathit{Apply}\ (f, \mathcal{E})]\!]\eta\varepsilon)\varepsilon$$ □

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 3. *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:*

$$\begin{array}{ll} & [\![\mathcal{E}_0\ (\sigma X(d_X{:}D_X) = f(X) \wedge \psi)\ \mathcal{E}_1]\!]\eta\varepsilon(Z)(\,[\![e]\!]\varepsilon) \\ = & [\![\mathit{Apply}\ (f, \mathcal{E}_0)\ (\sigma X(d_X{:}D_X) = \psi)\ \mathit{Apply}\ (f, \mathcal{E}_1)]\!]\eta\varepsilon(Z)(\,[\![e]\!]\varepsilon) \end{array}$$ □

Corollary 3 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 3. *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:*

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

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 3, it suffices to solve the following equation instead:

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

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

In the terminology of [15], equation (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 3 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 unfolding or substitutions are applied to the equation system. The latter is important, since this means that both manipulations aid in finding useful invariants.

Theorem 3. *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. □

An interesting observation that follows from Theorem 3 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 4. Consider the following two equation systems $(\mu X(n{:}\mathbb{N}) = X(n+1))$ and $(\nu X(n{:}\mathbb{N}) = X(n+1))$. Both equations have exactly the same set of invariants, since the invariant conditions of Definition 8 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 11. *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 $X, Y \in \mathsf{bnd}(\mathcal{E})$:*

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

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

$$
\begin{aligned}
& (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\}\} \\
& ((f(X) \wedge \phi_X[\phi_{Y\langle d_Y \rangle}/Y]) \\
& \qquad \left[_{Z\in V\setminus\{Y\}}(f(Z) \wedge Z(d_Z))_{\langle d_Z\rangle}/Z\right]\ ) \left[(f(Y) \wedge Y(d_Y))_{\langle d_Y\rangle}/Y\right] \\
\leftrightarrow\ & \{\text{distributivity of substitution over } \wedge,\ f \text{ is simple;}\} \\
& \{\text{Lemma 5 successively applied to all } Z \in V \setminus \{X\};\ \text{Lemma 2}\} \\
& (f(X) \wedge \phi_X \left[_{Z\in V\setminus\{Y\}}(f(Z) \wedge Z(d_Z))_{\langle d_Z\rangle}/Z\right] \\
& \qquad [\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])[(f(Y) \wedge Y(d_Y))_{\langle d_Y\rangle}/Y] \\
\leftrightarrow\ & \{\text{distributivity of substitution over } \wedge,\ f \text{ is simple;}\} \\
& \{\text{Lemma 4, } (V \setminus \{Y\}) \cup \{Y\} = V\} \\
& (f(X) \wedge \phi_X \left[_{Z\in V\setminus\{Y\}}(f(Z) \wedge Z(d_Z))_{\langle d_Z\rangle}/Z\right]\ ) \\
& \qquad [\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{Proposition 2, Lemma 2}\} \\
& (f(X) \wedge \phi_X\ [(f(Y) \wedge Y(d_Y))_{\langle d_Y\rangle}/Y]) \\
& \qquad [\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 4}\} \\
& (f(X) \wedge \phi_X)\ [(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] \\
\leftrightarrow\ & \{\text{Proposition 2, Lemma 2}\} \\
& (f(X) \wedge \phi_X)[(f(Y) \wedge \phi_{Y\langle d_Y \rangle})/Y] \\
\leftrightarrow\ & \{\text{Lemma 4}\} \\
& (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 2: } (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]
\end{aligned}
$$

□

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

Theorem 4. *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 11 by taking $Y = X$. □

The reverse of Theorem 4 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 5. Let $\nu X(n{:}\mathbb{N}) = X(n + 1)$ be an equation system. Using unfolding, we obtain the following equivalent equation system: $\nu X(n{:}\mathbb{N}) = X(n + 2)$. 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 5. *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 $\mathcal{E}'$ 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 11. □

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 5) 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 6. Consider the system $(\mu X(n{:}\mathbb{N}) = Y(n+1))$ $(\mu Y(n{:}\mathbb{N}) = X(2n))$. 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 $(\mu X(n{:}\mathbb{N}) = X(2(n+1)))$ $(\mu Y(n{:}\mathbb{N}) = X(2n))$, 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 of condition-action-effect rules. 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 parallism, 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 10. *A* linear process equation *is a parameterised equation taking the form*

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

where $f_{\mathsf{a}}{:}D \times E_{\mathsf{a}} \to D_{\mathsf{a}}$, $g_{\mathsf{a}}{:}D \times E_{\mathsf{a}} \to D$ *and* $c_{\mathsf{a}}{:}D \times E_{\mathsf{a}} \to \mathbb{B}$ *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 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 an action a carrying data parameter $f_{\mathsf{a}}(d, e_{\mathsf{a}})$ is possible 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 Def. 10.

Definition 11. *The* labelled transition system *of the LPE* P *of Def. 10 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,*
– $\rightarrow = \{(d, \mathsf{a}(v), d') \mid \mathsf{a} {\in} \mathcal{A}ct \wedge \exists e_{\mathsf{a}} {\in} E_{\mathsf{a}}.\ c_{\mathsf{a}}(d, e_{\mathsf{a}}) \wedge v = f_{\mathsf{a}}(d, e_{\mathsf{a}}) \wedge d' = g_{\mathsf{a}}(d, e_{\mathsf{a}})\}$ *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 $g_{\mathsf{a}}(d, e_{\mathsf{a}})$ that are reachable from d via enabled actions a. Invariants are useful for quickly verifying certain safety properties.

Definition 12. *A simple predicate* ι *is an* invariant *of* P *iff the following ordering holds for all actions* $\mathsf{a} \in \mathcal{A}ct$*:*

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

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

Example 7. To illustrate the notion of a process invariant, consider the following LPE:

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

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 [19, 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 [17]. 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:

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

where σ is a least or greatest fixed point sign, and $\oplus \in \{\wedge, \vee\}$ and $\mathsf{Q} \in \{\forall, \exists\}$ 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 $[\![\_]\!]^{\theta\varepsilon}_{P}$ 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 [19, 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 [19, 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 12. *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:*

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

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{array}{ll}
\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))\ \mathbf{E}(\phi) \\
\\
\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) & = \bigwedge_{\mathsf{a}\in\mathcal{A}ct} \forall e_{\mathsf{a}}{:}D_{\mathsf{a}}\ (c_{\mathsf{a}}(d,e_{\mathsf{a}}) \wedge \mathsf{match}(\mathsf{a}(f_{\mathsf{a}}(d,e_{\mathsf{a}})),\alpha)) \\
 & \qquad \implies (\mathbf{RHS}_\Phi(\phi)[g_{\mathsf{a}}(d,e_{\mathsf{a}})/d]) \\
\mathbf{RHS}_\Phi(\langle\alpha\rangle\phi) & = \bigvee_{\mathsf{a}\in\mathcal{A}ct} \exists e_{\mathsf{a}}{:}D_{\mathsf{a}}\ (c_{\mathsf{a}}(d,e_{\mathsf{a}}) \wedge \mathsf{match}(\mathsf{a}(f_{\mathsf{a}}(d,e_{\mathsf{a}})),\alpha) \\
 & \qquad \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)) \\
\\
\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) \\
\\
\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{array}
$$

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

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

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

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

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

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

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

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

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

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

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

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

$$
\begin{aligned}
& \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 \\
& \qquad (\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 \\
& \qquad (\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 \\
& \qquad (\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{aligned}
$$

– 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 6. *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 immediately from Lemma 12 and the translation $\mathbf{E}(\phi)$. □

Note that the reverse of Theorem 6 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 8. 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.

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

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

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

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, where $\mathcal{A}ct_\tau =_{def} \mathcal{A}ct \cup \{\tau\}$:

$$
\begin{aligned}
M(d{:}D^M) &= \sum\{ \sum_{e_{\mathsf{a}}:E_{\mathsf{a}}^M} c_{\mathsf{a}}^M(d,e) \implies \mathsf{a}(f_{\mathsf{a}}^M(d,e)) \cdot M(g_{\mathsf{a}}^M(d,e)) \mid a \in \mathcal{A}ct_\tau\} \\
S(d{:}D^S) &= \sum\{ \sum_{e_{\mathsf{a}}:E_{\mathsf{a}}^S} c_{\mathsf{a}}^S(d,e) \implies \mathsf{a}(f_{\mathsf{a}}^S(d,e)) \cdot S(g_{\mathsf{a}}^S(d,e)) \mid \mathsf{a} \in \mathcal{A}ct_\tau\}
\end{aligned}
$$

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

Algorithm 1 Generation of a PBES for Strong Bisimulation

$\mathsf{sbisim} = \nu E$, **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 we use the following abbreviations, for all $\mathsf{a} \in Act \wedge (p,q) \in \{(M,S),(S,M)\}$:

$$\mathsf{match}^{p,q}(d{:}D^p, d'{:}D^q) = \bigwedge_{\mathsf{a}\in Act} \forall e{:}E_{\mathsf{a}}^p.(c_{\mathsf{a}}^p(d,e) \implies \mathsf{step}_{\mathsf{a}}^{p,q}(d,d',e));$$

$$
\begin{aligned}
&\mathsf{step}_{\mathsf{a}}^{p,q}(d{:}D^p, d'{:}D^q, e{:}E_{\mathsf{a}}^p) = \\
&\qquad \exists e'{:}E_{\mathsf{a}}^q.c_{\mathsf{a}}^q(d',e') \wedge (f_{\mathsf{a}}^p(d,e) = f_{\mathsf{a}}^q(d',e')) \wedge X^{p,q}(g_{\mathsf{a}}^p(d,e), g_{\mathsf{a}}^q(d',e'));
\end{aligned}
$$

Theorem 7. *Let ι be a process invariant for LPE M. Let f^M be the simple function defined as:*

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

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:

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

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

$$
\begin{aligned}
&(\iota \wedge \mathsf{match}^{M,S}(d,d'))\ [_{Z\in\{X^{M,S},X^{S,M}\}}(\iota \wedge Z(d_Z))_{\langle d_Z\rangle}/Z] \\
\leftrightarrow\ &\{\text{By definition of syntactic substitution}\} \\
&\iota \wedge \textstyle\bigwedge_{\mathsf{a}\in\mathcal{A}ct}\ \forall e{:}E_{\mathsf{a}}^M.\ (c_{\mathsf{a}}^M(d,e) \implies \\
&\quad \exists e'{:}E_{\mathsf{a}}^S.\ c_{\mathsf{a}}^S(d',e') \wedge f_{\mathsf{a}}^M(d,e) = f_{\mathsf{a}}^S(d',e') \wedge (\iota[g_{\mathsf{a}}^M(d,e)/d] \wedge X^{M,S}(g_{\mathsf{a}}^M(d,e), g_{\mathsf{a}}^S(d',e')))) \\
\leftrightarrow\ &\{\text{for all actions } \mathsf{a} \text{ and for all } e{:}E_{\mathsf{a}}^M{:}\ \iota \wedge c_{\mathsf{a}}^M(d,e) \text{ implies } \iota[g_{\mathsf{a}}^M(d,e)/d]\ \} \\
&\iota \wedge \textstyle\bigwedge_{\mathsf{a}\in\mathcal{A}ct}\ \forall e{:}E_{\mathsf{a}}^M.\ (c_{\mathsf{a}}^M(d,e) \implies \\
&\quad \exists e'{:}E_{\mathsf{a}}^S.\ c_{\mathsf{a}}^S(d',e') \wedge f_{\mathsf{a}}^M(d,e) = f_{\mathsf{a}}^S(d',e') \wedge X^{M,S}(g_{\mathsf{a}}^M(d,e), g_{\mathsf{a}}^S(d',e'))) \\
\leftrightarrow\ &\{\text{definition of } \mathsf{match}\} \\
&(\iota \wedge \mathsf{match}^{M,S}(d,d'))
\end{aligned}
$$

and, likewise:

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

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.

Algorithm 2 Generation of a PBES for Branching Bisimulation for LPEs M and S.

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

$$
\begin{array}{rl}
E_2 := & \{X^{M,S}(d:D^M, d':D^S) = \mathsf{match}^{M,S}(d,d') \wedge \mathsf{match}^{S,M}(d',d)\ , \\
& \ X^{S,M}(d':D^S, d:D^M) = X^{M,S}(d,d')\ \} \\
E_1 := & \{Y^{p,q}_{\mathsf{a}}(d:D^p, d':D^q, e:E^p_{\mathsf{a}}) = \mathsf{close}^{p,q}_{\mathsf{a}}(d,d',e) \\
& \quad \mid a \in \mathit{Act} \wedge (p,q) \in \{(M,S),(S,M)\}\}
\end{array}
$$

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

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

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

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

Theorem 8. *Let ι be a process invariant for LPE M. Let f^M be the simple function defined as:*

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

Then f^M is a global invariant of $\mathsf{brbisim}$.

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

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

For the first equivalence, we reason as follows:

$$
\begin{array}{l}
\quad (\iota \wedge \mathsf{match}^{M,S}(d,d') \wedge \mathsf{match}^{S,M}(d',d)) \left[_{Z\in V}(f^M(Z) \wedge Z(d_Z))_{\langle d_Z\rangle}/Z\right] \\
\leftrightarrow \{\text{Definition of } \mathsf{match} \text{ and syntactic substitution}\} \\
\quad \iota \wedge \bigwedge \mathsf{a} \in \mathit{Act} \forall e{:}E_{\mathsf{a}}^M.\ c_{\mathsf{a}}^M(d,e) \implies (c_{\mathsf{a}}^M(d,e) \wedge \iota \wedge Y_{\mathsf{a}}^{M,S}(d,d',e)) \\
\qquad \bigwedge \mathsf{a} \in \mathit{Act} \forall e{:}E_{\mathsf{a}}^M.\ c_{\mathsf{a}}^S(d',e) \implies (\iota \wedge Y_{\mathsf{a}}^{M,S}(d,d',e)) \\
\leftrightarrow \{\text{logic}\} \\
\quad \iota \wedge \bigwedge \mathsf{a} \in \mathit{Act} \forall e{:}E_{\mathsf{a}}^M.\ c_{\mathsf{a}}^M(d,e) \implies Y_{\mathsf{a}}^{M,S}(d,d',e) \\
\qquad \bigwedge \mathsf{a} \in \mathit{Act} \forall e{:}E_{\mathsf{a}}^M.\ c_{\mathsf{a}}^S(d',e) \implies Y_{\mathsf{a}}^{M,S}(d,d',e) \\
\leftrightarrow \{\text{Definition of } \mathsf{match}\} \\
\quad (\iota \wedge \mathsf{match}^{M,S}(d,d') \wedge \mathsf{match}^{S,M}(d',d))
\end{array}
$$

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

$$
\begin{array}{l}
\quad (c_{\mathsf{a}}^M(d,e) \wedge \iota \wedge \mathsf{close}_{\mathsf{a}}^{M,S}(d,d',e)) \left[_{Z\in V}(f^M(Z) \wedge Z(d_Z))_{\langle d_Z\rangle}/Z\right] \\
\leftrightarrow \{\text{Definition of } \mathsf{close} \text{ and syntactic substitution }\} \\
\quad (c_{\mathsf{a}}^M(d,e) \wedge \iota \wedge (\exists e'{:}E_\tau^S.\ (c_\tau^S(d',e') \wedge (c_{\mathsf{a}}^M(d,e) \wedge \iota \wedge Y_{\mathsf{a}}^{M,S}(d,g_\tau^S(d',e'),e))) \\
\qquad \vee((\iota \wedge X^{M,S}(d,d')) \wedge ((\mathsf{a}=\tau \wedge \iota[g_\tau^M(d,e)/d] \wedge X^{M,S}(g_\tau^M(d,e),d')) \\
\qquad\quad \vee \exists e'{:}E_{\mathsf{a}}^S.\ c_{\mathsf{a}}^S(d',e') \wedge f_{\mathsf{a}}^M(d,e) = f_{\mathsf{a}}^S(d',e') \wedge (\iota[g_{\mathsf{a}}^M(d,e)/d] \wedge X^{p,q}(g_{\mathsf{a}}^M(d,e),g_{\mathsf{a}}^S(d',e')))))) \\
\leftrightarrow \{c_{\mathsf{a}}^M(d,e) \wedge \iota \text{ implies } \iota[g_{\mathsf{a}}^M(d,e)/d] \text{ (likewise for } \mathsf{a}=\tau\text{); general logic}\} \\
\quad (c_{\mathsf{a}}^M(d,e) \wedge \iota \wedge (\exists e'{:}E_\tau^S.\ (c_\tau^S(d',e') \wedge Y_{\mathsf{a}}^{M,S}(d,g_\tau^S(d',e'),e)) \\
\qquad \vee(X^{M,S}(d,d') \wedge ((\mathsf{a}=\tau \wedge X^{M,S}(g_\tau^M(d,e),d')) \\
\qquad\quad \vee \exists e'{:}E_{\mathsf{a}}^S.\ c_{\mathsf{a}}^S(d',e') \wedge f_{\mathsf{a}}^M(d,e) = f_{\mathsf{a}}^S(d',e') \wedge X^{p,q}(g_{\mathsf{a}}^M(d,e),g_{\mathsf{a}}^S(d',e')))))) \\
\leftrightarrow \{\text{Definition of } \mathsf{close}\} \\
\quad c_{\mathsf{a}}^M(d,e) \wedge \iota \wedge \mathsf{close}_{\mathsf{a}}^{M,S}(d,d',e)
\end{array}
$$

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

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

Algorithm 3 Generation of a PBES for (Branching) Simulation Equivalence

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

$$
\begin{array}{ll}
E_2 := \{ & X(d{:}D^M, d'{:}D^S) = X^{M,S}(d,d') \wedge X^{S,M}(d',d), \\
& X^{M,S}(d{:}D^M, d'{:}D^S) = \mathsf{match}^{M,S}(d,d'), \\
& X^{S,M}(d'{:}D^S, d{:}D^M) = \mathsf{match}^{S,M}(d',d)\} \\
E_1 := \{ & Y_{\mathsf{a}}^{p,q}(d{:}D^p, d'{:}D^q, e{:}E_{\mathsf{a}}^p) = \mathsf{close}_{\mathsf{a}}^{p,q}(d,d',e) \mid \mathsf{a} \in \mathit{Act} \wedge (p,q) \in \{(M,S),(S,M)\}\}
\end{array}
$$

Where match and close are as defined in Alg. 2.

Theorem 9. *Let ι be a process invariant for LPE M. Let f^M be the simple function defined as:*

$$
f^M(Z) = \begin{cases} \iota & \text{if } Z \in \{X, X^{M,S}, X^{S,M}, Y_{\mathsf{a}}^{S,M} \mid \mathsf{a} \in \mathit{Act}_\tau\} \\ \iota \wedge c_{\mathsf{a}}^M(d,e) & \text{if } Z \in \{Y_{\mathsf{a}}^{M,S} \mid \mathsf{a} \in \mathit{Act}_\tau\} \end{cases}
$$

Then f^M is a global invariant of brsim.

Proof. The proof follows the same line of reasoning as the proof for Theorem 8, and is therefore omitted. ☐

Algorithm 4 Generation of a PBES for Weak Bisimulation

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

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

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

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

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

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

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

Theorem 10. *Let ι be a process invariant for LPE M. Let f^M be the simple function defined as:*

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

Then f^M is a global invariant of wbisim.

Proof. The proof follows the same line of reasoning as the proof for Theorem 7 and 8. ☐

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.

7 Examples

So far, we have studied the theory of invariants in equation systems. We next illustrate how the use of invariants can help solving equation systems that arise naturally when solving 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 deliberately 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.

7.1 Insertion sort

Insertion sort is a classical sorting algorithm. To prove its correctness in the PBES framework, we use the algorithm's known invariants, which can immediately be checked on the process model and used as PBES invariants as ensured by Theorem 6. 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 ($\mathcal{L}(\mathbb{N})$) and parameters i, j, k. 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 k, which has been set to $l.j$. As long as $l.i > k$, $l.i$ 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.

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

Partial correctness of the algorithm means that the list that is reported via action `stop` is sorted:

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

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 $I(l, j, i, k) \models \phi$ yields, the following equation system:

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

Note that we have the following equivalence:

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

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.

We next 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 and 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 indeces i and j are larger than the key k, and that, during the search phase, the upper end of the current sublist is also sorted, i.e. $l.(j-1) < l.j$.

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

The above formula can be checked to be an invariant of process I, and, by Theorem 6, it is therefore also an invariant for equation X. Moreover, we find:

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

As a consequence, the invariant-strengthened equation system can be brought into the form of Proposition 3, from which it follows that the invariant is the solution to the invariant-strengthened equation system. This means that for any system $I(l, i, j, k)$ with values for i, j and k, 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 (coallitions 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 $\mathtt{v}(i)$ 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.

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

One way to formalize 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 [4].

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

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 2, 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:

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

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 3, 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 demonstration 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:

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

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:

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

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:

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

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 invariant-strengthened equation for Y yields the following equivalent equation for Y:

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

Since $m \in q$ is a global invariant, we find that, as a consequence of Proposition 3, 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:

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

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.

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

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 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:

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

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 6 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 3 and simplify the equation for Y to the one below:

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

Substituting this solution for Y in X and using Proposition 2 to simplify the resulting equation, we find the following equivalent equation for X:

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

Another application of Proposition 3 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:

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

The equation system encoding the above property is the following:

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

Using Property 2, $r = n_r$ 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:

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

Proposition 3 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:

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

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:

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

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

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

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 posesses 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-strenghtened 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 3 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 [22] and one of its essential safety properties. The cache coherence protocol serves $N > 0$ clients, and has a single controlling component. Messages between the clients and the controller are sent via channels. The model we present below 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. 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:

$$\begin{array}{l}\nu X.\ [\top]X \wedge \forall m{:}\mathbb{N}.\ [\mathtt{exclusive}(m)](\nu Y.\ [\neg\mathtt{invalidate}(m)]Y \\ \qquad \wedge[\exists j{:}\mathbb{N}.\ (\mathtt{exclusive}(j) \vee \mathtt{shared}(j))]\bot)\end{array} \tag{10}$$

The above formula is a variation of what is checked in [1, 22]. 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.

$$
\begin{array}{ll}
\textbf{sort} & M_1 ::= \mathit{empty} \mid \mathit{req\_exclusive} \mid \mathit{req\_shared} \\
& M_2 ::= \mathit{empty} \mid \mathit{invalidate} \mid \mathit{grant\_exclusive} \mid \mathit{grant\_shared} \\
& M_3 ::= \mathit{empty} \mid \mathit{invalidate\_ack} \\
& S \;\; ::= \mathit{invalid} \mid \mathit{shared} \mid \mathit{exclusive}
\end{array}
$$

$$
\begin{array}{l}
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{:}\mathbb{N}, e{:}\mathbb{B}, i, s{:}\mathcal{L}(\mathbb{B}), \mathit{cd}{:}M_1) = \\
\quad \sum n{:}\mathbb{N}.\ (n \le N \wedge c_1.n = \mathit{empty} \wedge \mathit{cache}.n = \mathit{invalid}) \\
\qquad \implies \mathtt{req\_shared}(n) \cdot P(\mathit{cache} := \mathit{cache}[\mathit{invalid}/n], c_1 := c_1[\mathit{req\_shared}/n]) \\
+ \sum n{:}\mathbb{N}.\ (n \le N \wedge c_1.n = \mathit{empty} \wedge (\mathit{cache}.n = \mathit{invalid} \vee \mathit{cache}.n = \mathit{shared})) \\
\qquad \implies \mathtt{req\_exclusive}(n) \cdot P(c_1 := c_1[\mathit{req\_exclusive}/n]) \\
+ \sum n{:}\mathbb{N}.\ (n \le N \wedge c_2.n = \mathit{invalidate} \wedge c_3.n = \mathit{empty}) \\
\qquad \implies \mathtt{invalidate\_ack}(n) \cdot \\
\qquad\quad P(\mathit{cache} := \mathit{cache}[\mathit{invalid}/n], c_2 := c_2[\mathit{empty}/n], c_3 := c_3[\mathit{invalidate\_ack}/n]) \\
+ \sum n{:}\mathbb{N}.\ (n \le N \wedge c_2.n = \mathit{grant\_shared}) \\
\qquad \implies \mathtt{shared}(n) \cdot P(\mathit{cache} := \mathit{cache}[\mathit{shared}/n], c_2 := c_2[\mathit{empty}/n]) \\
+ \sum n{:}\mathbb{N}.\ (n \le N \wedge c_2.n = \mathit{grant\_exclusive}) \\
\qquad \implies \mathtt{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 \implies \mathtt{grant\_shared}(k) \cdot P(c_2 := c_2[\mathit{grant\_shared}/k], 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 \implies \mathtt{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}) \\
+ \sum n{:}\mathbb{N}.\ (n \le N \wedge \mathit{cd} = \mathit{empty} \wedge c_1.n \ne \mathit{empty}) \\
\qquad \implies \tau \cdot P(c_1 := c_1[\mathit{empty}/n], k := n, i := s, \mathit{cd} := c_1.n) \\
+ \sum n{:}\mathbb{N}.\ (n \le N \wedge i.n \wedge c_2.n = \mathit{empty} \wedge (\mathit{cd} = \mathit{req\_exclusive} \vee (\mathit{cd} = \mathit{req\_shared} \wedge e))) \\
\qquad \implies \mathtt{invalidate}(n) \cdot P(c_2 := c_2[\mathit{invalidate}/n], i := i[\bot/n]) \\
+ \sum n{:}\mathbb{N}.\ (n \le N \wedge \mathit{cd} \ne \mathit{empty} \wedge c_3.n = \mathit{invalidate\_ack}) \\
\qquad \implies \tau \cdot P(c_3 := c_3[\mathit{empty}/n], k, e := \bot, s := s[\bot/n])
\end{array}
$$

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

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, predicate 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 $[\exists j{:}\mathbb{N}.\ (\mathtt{exclusive}(j) \vee \mathtt{shared}(j))]\bot$ 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, $\forall n{:}\mathbb{N}.\ (n \le N \implies (c_2.n \ne \mathit{grant\_shared} \wedge c_2.n \ne \mathit{grant\_exclusive}))$. So, we are looking for an invariant that overapproximates this last line and hopefully still characterises all states reachable from an instantiation of Y as occuring 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:

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

α satisifes the sufficient condition of Property 3 for the equation of Y and is therefore a local invariant for Y. Since α implies $\forall n{:}\mathbb{N}.\ (n \le N \implies (c_2.n \ne \mathit{grant\_shared} \wedge c_2.n \ne \mathit{grant\_exclusive}))$, the equation of Y after strengthening with α has the same form as before strengthening, except the

Table 3. equation system

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

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

last line changes into α. By applying Proposition 3, 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 3, 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,

$$
\begin{aligned}
(\lambda)\ :\ & \forall n{:}\mathbb{N}.(n \leq N \wedge c_2.n = \mathit{grant\_exclusive}) \\
& \implies e \wedge s.n \wedge \forall j{:}\mathbb{N}.\ (j \leq N \wedge j \neq n \implies (c_2.j = \mathit{empty} \wedge i.j = \bot)) \\
& \quad \wedge(\mathit{cd} = \mathit{empty} \vee \forall j{:}\mathbb{N}.\ (j \leq N \implies (c_3.j = \mathit{empty})))
\end{aligned}
$$

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

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

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 6, it is also a local invariant for both X and Y. Let us take as β the predicate $\kappa \wedge (\exists n{:}\mathbb{N}.\ (n \leq N \implies c_2.n = \mathit{grant\_exclusive}) \implies e)$. By verifying the assumptions of Property 3 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 $(\exists n{:}\mathbb{N}.\ (n \leq N \implies c_2.n = \mathit{grant\_exclusive}) \implies e)$ is preserved by the summand of X's equation where the guard is $\mathit{cd} \neq \mathit{empty} \wedge c_3.n = \mathit{invalidate\_ack}$ and, in the right-hand side parameter list, $e = \bot$. We reason as follows: Suppose $\exists j{:}\mathbb{N}.\ (j \leq N \implies c_2.j = \mathit{grant\_exclusive})$ and e are true beforehand. Then it follows from $(\kappa 5)$ that $s.j = \top$. Note also that $c_3.n = \mathit{invalidate\_ack}$ implies (due to invariant $\kappa 5$) $s.n = \top$. So, from $e = \top$ and $(\kappa 1)$ follows that $j = n$, meaning that $c_2.n = \mathit{grant\_exclusive}$, which, via $(\kappa 4)$, implies $c_3.n = \mathit{empty}$, contradicting the guard $c_3.n = \mathit{invalidate\_ack}$. Therefore, $e = \bot$ must hold already in the guard, and since c_2 is not modified, the implication $(\exists n{:}\mathbb{N}.\ (n \leq N \implies c_2.n = \mathit{grant\_exclusive}) \implies e)$ does not change its truth value.

We proceed to solve the PBES strenghtened 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-strenghtened) equation of X, also that β is a solution for X. For both equations, we made use of Proposition 3.

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

$$
\begin{aligned}
&\beta(\mathit{invalid} \ldots \mathit{invalid}, \mathit{empty} \ldots \mathit{empty}, \mathit{empty} \ldots \mathit{empty}, \mathit{empty} \ldots \mathit{empty}, \\
&\quad 0, \bot, \bot \ldots \bot, \bot \ldots \bot, \mathit{empty}) = \top
\end{aligned}
$$

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 conjecture that many such techniques, see e.g. [21, 22], can be put to use for (automatically) discovering global invariants in PBESs. Additional research is of course needed to substantiate this conjecture.

Acknowledgements
The authors would like to thank Jan Friso Groote for providing a counterexample (see Example 2) to a conjecture involving syntactic substition. 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, and K. Stahl. Parameterized verification of a cache coherence protocol: Safety and liveness. In *VMCAI*, pages 317–330, 2002.
2. M.A. Bezem and J.F. Groote. Invariants in process algebra with data. In *Proceedings Concur'94*, volume 836 of *LNCS*, pages 401–416. Springer Verlag, 1994.
3. A.R. Bradley, Z. Manna, and H.B. Sipma. The polyranking principle. In *ICALP*, pages 1349–1361, 2005.
4. T. Chen, B. Ploeger, J. van de Pol, and T.A.C. Willemse. Equivalence checking for infinite systems using parameterized boolean equation systems. In *Proc. 18th Int'l Conference on CONCUR*, LNCS 4703, pages 120–135. Springer, 2007.
5. A. van Dam, B. Ploeger, and T.A.C. Willemse. Instantiation for parameterised boolean equation systems. In *Proceedings of ICTAC'08*, 2008. To appear.
6. Y. Fang, N. Piterman, A. Pnueli, and L.D. Zuck. Liveness with invisible ranking. *STTT*, 8(3):261–279, 2006.
7. R.W. Floyd. Assigning meanings to programs. In *Mathematical Aspects of Computer Science, Proceedings of Symposia in Applied Mathematics 19*, pages 19–32. American Mathematical Society, 1967.
8. A. Fujioka, T. Okamoto, and K. Ohta. A practical secret voting scheme for large scale elections. In *Proc. 3rd Workshop on the Theory and Application of Cryptographic Techniques*, volume 718 of *LNCS*, 1992.
9. M.M. Gallardo, C. Joubert, and P. Merino. Implementing influence analysis using parameterised boolean equation systems. In *Proceedings of ISOLA'06*. IEEE Computer Society Press, November 2006.
10. H. Garavel, R. Mateescu, F. Lang, and W. Serwe. CADP 2006: A toolbox for the construction and analysis of distributed processes. In *Proceedings of CAV*, volume 4590 of *LNCS*, pages 158–163. Springer, 2007.
11. J.F. Groote and R. Mateescu. Verification of temporal properties of processes in a setting with data. In *Proc. of AMAST*, LNCS 1548, pages 74–90. Springer, 1999.

12. J.F. Groote, A.H.J. Mathijssen, M.A. Reniers, Y.S. Usenko, and M.J. van Weerdenburg. The formal specification language mCRL2. In *Methods for Modelling Software Systems (MMOSS)*, volume 06351 of *Dagstuhl Seminar Proceedings*, 2007.
13. J.F. Groote and A. Ponse. The syntax and semantics of mCRL. In *Algebra of Communicating Processes, Workshops in Computing*, pages 26–62, 1994.
14. J.F. Groote and T.A.C. Willemse. Model-checking processes with data. *Sci. Comput. Program*, 56(3):251–273, 2005.
15. J.F. Groote and T.A.C. Willemse. Parameterised boolean equation systems. *Theor. Comput. Sci*, 343(3):332–369, 2005.
16. C.A.R. Hoare. An axiomatic basis for computer programming. *Communications of the ACM*, 12(10):576–585, 1969.
17. D. Kozen. Results on the propositional mu-calculus. *Theor. Comp. Sci*, 27:333–354, 1983.
18. A. Mader. *Verification of Modal Properties Using Boolean Equation Systems*. PhD thesis, Technische Universität München, 1997.
19. R. Mateescu. Local model-checking of an alternation-free value-based modal mu-calculus. In *Proc. 2nd Int'l Workshop on VMCAI*, September 1998.
20. R. Mateescu. *Vérification des propriétés temporelles des programmes paralléles*. PhD thesis, Institut National Polytechnique de Grenoble, 1998.
21. S. Pandav, K. Slind, and G. Gopalakrishnan. Counterexample guided invariant discovery for parameterized cache coherence verification. In *Proceedings CHARME*, volume 3725 of *LNCS*, pages 317–331. Springer, 2005.
22. A. Pnueli, S. Ruah, and L. Zuck. Automatic deductive verification with invisible invariants. In *Proceedings TACAS*, volume 2031 of *LNCS*, pages 82–97, 2001.
23. S. Sankaranarayanan, H.B. Sipma, and Z. Manna. Constructing invariants for hybrid systems. *Formal Methods in System Design*, 32(1):25–55, 2008.
24. A. Tarski. A lattice-theoretical fixpoint theorem and its applications. *Pacific J. Mathematics*, 5(2):285–309, June 1955.
25. J.C. van de Pol, 2007. Personal communication.
26. D. Zhang and R. Cleaveland. Fast generic model-checking for data-based systems. In F. Wang, editor, *FORTE*, volume 3731 of *LNCS*, pages 83–97. Springer, 2005.