Instantiation for Parameterised Boolean Equation Systems

A. van Dam, B. Ploeger, and T.A.C. Willemse

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

Abstract. Verification problems for finite- and infinite-state processes, like model checking and equivalence checking, can effectively be encoded in Parameterised Boolean Equation Systems (PBESs). Solving the PBES solves the encoded problem. The decidability of solving a PBES depends on the data sorts that occur in the PBES. We describe a manipulation for transforming a given PBES to a simpler PBES that may admit solution methods that are not applicable to the original one. Depending on whether the data sorts occurring in the PBES are finite or countable, the resulting PBES can be a Boolean Equation System (BES) or an Infinite Boolean Equation System (IBES). Computing the solution to a BES is decidable. Computing the global solution to an IBES is still undecidable, but for partial solutions (which suffices for *e.g.* local model checking), effective tooling is possible. We give examples that illustrate the efficacy of our techniques.

1 Introduction

Parameterised Boolean Equation Systems (PBESs) [7, 10] have emerged as a versatile vehicle for studying and solving verification problems for complex systems. Prime examples are the encoding of the first-order modal μ-calculus model-checking problem over (possibly infinite) labelled transition systems [5, 6]; equivalence checking of various bisimulations on (possibly infinite) labelled transition systems [3]; and static analysis of code [4]. The solution to the encoded problem can be found by solving the resulting PBES. Solving PBESs is, much like the problems that can be encoded in them, generally undecidable. The outlook, however, is not that bleak: practical applications have illustrated that a pragmatic approach can lead to promising results [6]. Among the techniques for solving PBESs are *symbolic approximation* [6] and *pattern matching* [7].

We here report on techniques for partially and fully *instantiating* a PBES. In general, this results in a new PBES. Ultimately, the transformation can yield a *Boolean Equation System* (BES) [8] when all involved data sorts are finite, or an *Infinite Boolean Equation System* (IBES) [9] when all data sorts are countable. From a theoretical viewpoint, our transformations firmly relate the abovementioned different formalisms; moreover, the transformations help in understanding the quite complex theory underlying PBESs. In particular, they confirm the

intuition that the inherent computational complexity in PBESs is due to the complexity of the data sorts that appear in the equations.

On a more practical level, our technique for *partially* instantiating a PBES leads to a wider applicability of existing solution techniques — such as the aforementioned use of *pattern matching* — even in the presence of countable and uncountable data sorts. A full instantiation of PBESs to BESs allows for full automation: BESs form a subset of PBESs for which computing the solution is effectively decidable. As such, both types of instantiation are welcome additions to the library of PBES manipulation methods.

We illustrate the efficacy of our instantiation techniques using several examples. Partial instantiation is demonstrated to be very effective on a model checking example on an infinite state system from the literature; the example contains both finite and infinite data sorts, and partial instantiation enables the use of pattern matching for solving the problem. The effectiveness of fully instantiating a PBES to (I)BES is illustrated by a prototype tool that implements this instantiation. We report on our findings using it to solve equivalence checking and local model checking problems, encoded using PBESs.

Our transformations take inspiration from the algorithm that is proposed in [10] to construct *and* solve an alternation-free PBES —encoding the local model checking problem for the alternation-free modal μ-calculus— in an on-the-fly manner. We are not aware of any tool that actually implements the algorithm, but the experiments with our prototype implementation suggest the approach is practically feasible.

This paper is structured as follows. In Section 2, we introduce the relevant concepts and properties. Partial and full instantiation for finite data sorts in PBESs is detailed in Section 3, and full instantiation to IBESs for PBESs involving countably infinite data sorts is discussed in Section 4. In Section 5, we demonstrate the power of all involved instantiation schemes and our concluding remarks can be found in Section 6.

2 Preliminaries

2.1 Data

We assume that there are nonempty data sorts, generally written using letters D, E and F. We assume that every countable sort has a collection of *basic elements* to which every term can be rewritten. For a sort D, we write $v \in D$ to denote that v is a basic element of D and we use set notation to list the basic elements of D, *e.g.* $D = \{v_1, \ldots, v_n\}$. With every sort D we associate a semantic set $\mathbb{D}$ such that every syntactic term of type D can be mapped to the element of $\mathbb{D}$ it represents. The set of basic elements of a countable sort D is isomorphic to the semantic set $\mathbb{D}$.

We have a set $\mathcal{D}$ of *data variables*, with typical elements $d, d_1, \ldots$, and we assume that there is some data language that is sufficiently rich to denote all relevant *data terms*, such as for instance $3 + d_1 \leq d_2$. For a closed term t of type

D (denoted $t{:}D$), we assume an interpretation function $[\![t]\!]$ that maps t to the data element of $\mathbb{D}$ it represents. For open terms we use a *data environment* ε that maps each variable from $\mathcal{D}$ to a data element of the right sort. The interpretation of an open term t, denoted as $[\![t]\!]\varepsilon$ is given by $\varepsilon(t)$ where ε is extended to terms in the standard way.

We assume the existence of a sort $B = \{\top, \bot\}$ representing the Booleans $\mathbb{B}$, and a sort $N = \{0, 1, \ldots\}$ representing the natural numbers $\mathbb{N}$. For these sorts, we assume the usual operators are available and we do not write constants or operators in the syntactic domain any different from their semantic counterparts. For example, we have $\mathbb{B} = \{\top, \bot\}$ and the syntactic operator $\_ \wedge \_{:}B \times B \to B$ corresponds to the usual, semantic conjunction $\_ \wedge \_{:}\mathbb{B} \times \mathbb{B} \to \mathbb{B}$.

2.2 Parameterised Boolean equation systems

We are interested in solving sequences of fixpoint equations, where each equation is of the following form:

$$\sigma X(d_1{:}D_1, \ldots, d_n{:}D_n) = \varphi.$$

The left-hand side of each equation consists of a *fixpoint symbol* $\sigma \in \{\mu, \nu\}$, where μ indicates a least and ν a greatest fixpoint, and a *predicate variable* $X{:}D_1 \times \ldots \times D_n \to B$ (from a set of variables $\mathcal{X}$) that depends on n data variables $d_1, \ldots, d_n$ of possibly infinite sorts $D_1, \ldots, D_n$. If $n = 0$ we have $X{:}B$ and we call X a *proposition variable.*The right-hand side of each equation is a *predicate formula* containing data terms, Boolean connectives, quantifiers over (possibly infinite) data domains and data and predicate variables. The restriction to predicate variables of arity 1 or 2 does not incur a loss of generality, which is why we prefer such restricted forms in the theoretical considerations that follow.

Definition 1 (Predicate Formulae). Predicate formulae φ *are defined by the following grammar:*

$$\varphi ::= b \mid X(e) \mid \varphi \oplus \varphi \mid \mathsf{Q}d{:}D.\varphi$$

where $\oplus \in \{\wedge, \vee\}$, $\mathsf{Q} \in \{\forall, \exists\}$, b *is a data term of sort* B, X *is a predicate variable,* d *is a data variable of sort* D *and* e *is a data term.*

Note that negation does not occur in predicate formulae, except as an operator in data terms. As a notational convenience, we use the operators $\oplus$ and Q throughout this paper when the exact operator is of lesser importance. Also, we call a predicate formula φ *closed* if no data variable in φ occurs freely. We now formalise the notion of a *Parameterised Boolean Equation System.*

Definition 2 (Parameterised Boolean Equation System). *A* parameterised Boolean equation system (PBES) *is inductively defined as follows:*

- ϵ *is the empty PBES;*

– *for every PBES* $\mathcal{E}$, $(\sigma X(d{:}D) = \varphi)\ \mathcal{E}$ *is also a PBES, where* $\sigma{\in}\{\mu,\nu\}$ *is a fixpoint symbol,* $X{:}D \to B$ *is a predicate variable and* φ *is a predicate formula.*

A special class of PBESs is the class of *Boolean Equation Systems* (BESs). BESs have been studied extensively in the literature [8]; due to the absence of data expressions and first order constructs, and due to the use of proposition variables rather than predicate variables, BESs are easier to understand, as the underlying lattice is simpler. Formally, we have:

Definition 3 (Boolean Equation System). *A* Boolean equation system *is a PBES in which every predicate variable is of type* B *and every formula* φ *adheres to the following grammar (hereafter referred to as* proposition formulae*):*

$$\varphi ::= \top \mid \bot \mid X \mid \varphi \oplus \varphi$$

where $\oplus \in \{\wedge, \vee\}$ *and* X *is a proposition variable.*

The set of predicate variables that occur in a predicate formula φ, denoted by $\mathsf{occ}(\varphi)$, is defined recursively as follows, for any formulae φ_1, φ_2:

$$\mathsf{occ}(b) \triangleq \emptyset \qquad \mathsf{occ}(X(e)) \triangleq \{X\}$$
$$\mathsf{occ}(\varphi_1 \oplus \varphi_2) \triangleq \mathsf{occ}(\varphi_1) \cup \mathsf{occ}(\varphi_2) \qquad \mathsf{occ}(\mathsf{Q}d{:}D.\varphi_1) \triangleq \mathsf{occ}(\varphi_1).$$

For any PBES $\mathcal{E}$, the set of *binding predicate variables*, $\mathsf{bnd}(\mathcal{E})$, is the set of variables occurring at the left-hand side of some equation in $\mathcal{E}$. The set of *occurring predicate variables*, $\mathsf{occ}(\mathcal{E})$, is the set of variables occurring at the right-hand side of some equation in $\mathcal{E}$. The set of predicate variables occurring anywhere in $\mathcal{E}$ is denoted by $\mathsf{var}(\mathcal{E})$. Formally, we define:

$$\mathsf{bnd}(\epsilon) \triangleq \emptyset \qquad \mathsf{bnd}((\sigma X(d{:}D) = \varphi)\ \mathcal{E}) \triangleq \mathsf{bnd}(\mathcal{E}) \cup \{X\}$$
$$\mathsf{occ}(\epsilon) \triangleq \emptyset \qquad \mathsf{occ}((\sigma X(d{:}D) = \varphi)\ \mathcal{E}) \triangleq \mathsf{occ}(\mathcal{E}) \cup \mathsf{occ}(\varphi)$$
$$\mathsf{var}(\mathcal{E}) \triangleq \mathsf{bnd}(\mathcal{E}) \cup \mathsf{occ}(\mathcal{E}).$$

A PBES $\mathcal{E}$ is said to be *well-formed* iff every binding predicate variable occurs at the left-hand side of precisely one equation of $\mathcal{E}$. Thus, $(\nu X = \top)(\mu X = \bot)$ is not a well-formed PBES. We only consider well-formed PBESs in this paper.

We say a PBES $\mathcal{E}$ is *closed* whenever $\mathsf{occ}(\mathcal{E}) \subseteq \mathsf{bnd}(\mathcal{E})$ and if $\mathcal{E}$ is not closed, we say $\mathcal{E}$ is *open*. An equation $\sigma X(d{:}D) = \varphi$ is called *data-closed* if the set of data variables that occur freely in φ is either empty or $\{d\}$. A PBES is called *data-closed* iff each of its equations is data-closed.

We say an equation $\sigma X(d{:}D) = \varphi$ is *solved* if $\mathsf{occ}(\varphi) = \emptyset$, and a PBES $\mathcal{E}$ is *solved* iff each of its equations is solved.

Finally, we give the denotational semantics of predicate formulae and PBESs. Predicate formulae are interpreted in a context of a data environment ε and a *predicate environment* $\eta{:}\mathcal{X} \to (\mathbb{D} \to \mathbb{B})$. For an arbitrary environment θ (be it a

data environment or predicate environment), we write $\theta[v/d]$ for the environment θ in which the variable d has been assigned the value v. For substitution on tuples we define $\theta[(v_1,\ldots,v_n)/(d_1,\ldots,d_n)]$ to be equivalent to the simultaneous substitution $\theta[v_1/d_1,\ldots,v_n/d_n]$.

Definition 4 (Semantics of Predicate Formulae). *Let ε be a data environment and $\eta{:}\mathcal{X} \to (\mathbb{D} \to \mathbb{B})$ be a predicate environment. The* interpretation $[\![\varphi]\!]\eta\varepsilon$ *that maps a predicate formula φ to $\top$ or $\bot$, is inductively defined as follows:*

$$\begin{array}{ll} [\![b]\!]\eta\varepsilon & \stackrel{\Delta}{=} [\![b]\!]\varepsilon \\ [\![X(e)]\!]\eta\varepsilon & \stackrel{\Delta}{=} \eta(X)([\![e]\!]\varepsilon) \\ [\![\varphi_1 \oplus \varphi_2]\!]\eta\varepsilon & \stackrel{\Delta}{=} [\![\varphi_1]\!]\eta\varepsilon \oplus [\![\varphi_2]\!]\eta\varepsilon \\ [\![\mathsf{Q}d{:}D.\varphi]\!]\eta\varepsilon & \stackrel{\Delta}{=} \mathsf{Q}v{\in}\mathbb{D} : [\![\varphi]\!]\eta(\varepsilon[v/d]). \end{array}$$

The predicate formula φ in an equation $\sigma X(d{:}D) = \varphi$ must be interpreted as a fixpoint over the set of functions with domain $\mathbb{D}$ and co-domain $\mathbb{B}$. Note that the existence of such fixpoints follows from the following observations. The variable d, which may occur free in φ, is effectively used as a formal, syntactic function parameter. Semantically, this is achieved by associating the interpretation of the predicate formula φ to the functional $(\lambda v{\in}\mathbb{D}.\ [\![\varphi]\!]\eta\varepsilon[v/d])$, which relies on the data environment to assign specific values to variable d. The set of (total) functions $f{:}\mathbb{D} \to \mathbb{B}$, denoted by $\mathbb{B}^{\mathbb{D}}$ can be equipped with an ordering $\sqsubseteq$, defined as follows:

$$f \sqsubseteq g \ \stackrel{\Delta}{=}\ \forall d \in \mathbb{D} : f(d) \Rightarrow g(d).$$

The set $(\mathbb{B}^{\mathbb{D}}, \sqsubseteq)$ is a complete lattice. The functional $(\lambda v{\in}\mathbb{D}.\ [\![\varphi]\!]\eta\varepsilon[v/d])$ can be turned into a predicate formula transformer by employing the predicate environment η in a similar manner as the data environment is used to turn a predicate formula into a functional. Assuming that the domain of the predicate variable X is of sort D, the functional $(\lambda v{\in}\mathbb{D}.\ [\![\varphi]\!]\eta\varepsilon[v/d])$ yields the following predicate formula transformer:

$$\lambda g{\in}\mathbb{B}^{\mathbb{D}}.\ (\lambda v{\in}\mathbb{D}.\ [\![\varphi]\!]\eta[g/X]\varepsilon[v/d]).$$

The resulting predicate formula transformer is monotone over the complete lattice $(\mathbb{B}^{\mathbb{D}}, \sqsubseteq)$. As a corollary of Tarski's fixpoint Theorem [11], the existence of least and greatest fixpoints of the predicate formula transformers is guaranteed. We denote the extremal fixpoints of the above predicate formula transformers as follows:

$$\sigma g{\in}\mathbb{B}^{\mathbb{D}}.\ (\lambda v{\in}\mathbb{D}.\ [\![\varphi]\!]\eta[g/X]\varepsilon[v/d]).$$

Definition 5 (Solution of a PBES). *The* solution of a PBES *in the context of a predicate environment η and a data environment ε is inductively defined as follows, for any PBES $\mathcal{E}$:*

$$\begin{array}{ll} [\![\epsilon]\!]\eta\varepsilon & \stackrel{\Delta}{=}\ \eta \\ [\![(\sigma X(d{:}D) = \varphi)\mathcal{E}]\!]\eta\varepsilon & \stackrel{\Delta}{=}\ [\![\mathcal{E}]\!](\eta[\sigma f \in \mathbb{B}^{\mathbb{D}}.\ \lambda v{\in}\mathbb{D}.[\![\varphi]\!]([\![\mathcal{E}]\!]\eta[f/X]\varepsilon)\varepsilon[v/d]/X])\varepsilon. \end{array}$$

The solution of a PBES prioritises the fixpoint signs of equations that come first over the signs of equations that follow. In that sense, the solution is sensitive to the order of equations in a PBES. Moreover, the solution of a PBES only assigns functions to the *binding* variables of that PBES; other predicate variables are left unmodified. This follows from the following lemma:

Lemma 1. *Let $\mathcal{E}$ be an arbitrary PBES. Then:*

$$\forall X \notin \mathsf{bnd}(\mathcal{E}) : \ \forall \eta, \varepsilon : \ [\![\mathcal{E}]\!]\eta\varepsilon(X) = \eta(X).$$

Proof. The proof is by induction on the length of equation system $\mathcal{E}$. If $\mathcal{E}$ is of length 0 then the equivalence holds vacuously. Suppose $\mathcal{E}$ is of length $m+1$, and for all $\mathcal{E}'$ of length m, we have:

$$\text{(IH)} \qquad \forall X \notin \mathsf{bnd}(\mathcal{E}') : \ \forall \eta, \varepsilon : \ [\![\mathcal{E}']\!]\eta\varepsilon(X) = \eta(X).$$

Necessarily, $\mathcal{E}$ is of the form $(\sigma Z(f{:}F) = \varphi)\ \mathcal{F}$, where $\mathcal{F}$ is of length m. Let $X \notin \mathsf{bnd}((\sigma Z(f{:}F) = \varphi)\ \mathcal{F})$, and let η, ε be arbitrary environments. We derive:

$$\begin{array}{ll} & [\![(\sigma Z(f{:}F) = \varphi)\ \mathcal{F}]\!]\eta\varepsilon(X) \\ = & ([\![\mathcal{F}]\!]\eta[(\sigma g{\in}\mathbb{B}^{\mathbb{F}}.\ \lambda v{\in}\mathbb{F}.\ [\![\varphi]\!]([\![\mathcal{F}]\!]\eta[g/Z]\varepsilon)\varepsilon[v/f])/Z]\varepsilon)(X) \\ \overset{\text{(IH)}}{=} & (\eta[(\sigma g{\in}\mathbb{B}^{\mathbb{F}}.\ \lambda v{\in}\mathbb{F}.\ [\![\varphi]\!]([\![\mathcal{F}]\!]\eta[g/Z]\varepsilon)\varepsilon[v/f])/Z])(X) \\ \overset{X \neq Z}{=} & \eta(X). \end{array}$$

□

2.3 Infinite Boolean Equation Systems

Mader [8] introduces *Infinite* Boolean Equation Systems *(IBESs)* as a vehicle for solving a model checking problem for infinite state systems. IBESs resemble BESs but differ in the following aspects: (1) finite *and* (countably) infinite conjunction and disjunction over proposition variables are allowed, and (2) finite *and* (countably) infinite sequences of equations are allowed (but still only finitely many *blocks* of equations).

Definition 6 (Infinite Proposition Formulae). Infinite proposition formulae ω *are defined by the following grammar, for any countable sorts I and $J \subseteq I$:*

$$\omega ::= \top \mid \bot \mid X_i \mid \omega \oplus \omega \mid \bigoplus_{j \in J} \omega$$

where $\oplus \in \{\wedge, \vee\}$, $\bigoplus \in \{\bigwedge, \bigvee\}$ and $X_i{:}B$ is a proposition variable for any $i \in I$.

Here, $\bigwedge_{j \in J}$ and $\bigvee_{j \in J}$ denote the infinite conjunction and disjunction over basic elements of a countable sort J, respectively.

Definition 7 (Infinite Boolean Equation System). *An* infinite Boolean equation system *(IBES) is inductively defined as follows:*

- *ϵ is the empty IBES;*

– *for every IBES $\mathcal{E}$, $\sigma\mathcal{B}\ \mathcal{E}$ is also an IBES, where $\sigma \in \{\mu, \nu\}$ is a fixpoint symbol and $\sigma\mathcal{B}$ is a block of equations $\{\sigma X_j = \omega_j \mid j{\in}J\}$ where J is a countable sort, and for each $j{\in}J$, $X_j{:}B$ is a proposition variable and ω_j is an infinite proposition formula.*

Notice that BESs are, syntactically, exactly in the intersection of PBESs and IBESs. The notions of binding and occurring variables, and the induced notions of *open, closed* and *well-formedness*, that are defined for PBESs transfer to IBESs without problems. We also restrict to IBESs that are well-formed.

The semantics of infinite proposition formulae is defined in the context of a proposition environment $\eta{:}\mathcal{X} \to \mathbb{B}$. For any countable sort I, environment η and function $f{:}I \to \mathbb{B}$ we denote by $\eta[f/X_I]$ the simultaneous substitution of $f(i)$ for X_i in η for all $i{\in}I$, such that $\eta[f/X_I](X_i) = f(i)$ if $i{\in}I$ and $\eta(X_i)$ otherwise.

Definition 8 (Semantics of Infinite Proposition Formulae). *Let $\eta{:}\mathcal{X} \to \mathbb{B}$ be a proposition environment. The* interpretation $[\![\omega]\!]\eta$ *that maps an infinite proposition formula ω to $\top$ or $\bot$, is inductively defined as follows:*

$$\begin{array}{ll} [\![\top]\!]\eta & \stackrel{\Delta}{=} \top \\ [\![\bot]\!]\eta & \stackrel{\Delta}{=} \bot \\ [\![X_i]\!]\eta & \stackrel{\Delta}{=} \eta(X_i) \\ [\![\omega_1 \oplus \omega_2]\!]\eta & \stackrel{\Delta}{=} [\![\omega_1]\!]\eta \oplus [\![\omega_2]\!]\eta \\ [\![\bigoplus_{j\in J} \omega]\!]\eta & \stackrel{\Delta}{=} \mathsf{Q}v{\in}J : [\![\omega[v/j]]\!]\eta \end{array}$$

where $\mathsf{Q} = \forall$ if $\bigoplus = \bigwedge$, and $\mathsf{Q} = \exists$ otherwise.

The set of functions $f{:}I \to \mathbb{B}$, where I is some (countable) sort, is denoted by $\mathbb{B}^I$. Together with the ordering $\sqsubseteq$, the set $(\mathbb{B}^I, \sqsubseteq)$ is a complete lattice. Let $\Omega = \{\omega_i \mid i \in I\}$ be a countable set of infinite proposition formulae. The functional induced by the interpretation of Ω is written $(\lambda i{\in}I.\ [\![\omega_i]\!]\eta)$, with $\omega_i \in \Omega$. This leads to the following transformer on infinite proposition formulae:

$$\lambda g \in \mathbb{B}^I.\ (\lambda i{\in}I.\ [\![\omega_i]\!]\eta[g/X_I]).$$

The transformer is a monotone operator on the complete lattice $(\mathbb{B}^I, \sqsubseteq)$, guaranteeing the existence of its least and greatest fixpoints.

Definition 9 (Solution of an Infinite BES). *Let η be a proposition environment, $\mathcal{E}$ be an IBES and $\sigma\mathcal{B} = \{\sigma X_i = \omega_i \mid i{\in}I\}$ be a block for some countable sort I. The* solution of an IBES *is inductively defined as follows:*

$$\begin{array}{ll} [\![\epsilon]\!]\eta & \stackrel{\Delta}{=} \eta \\ [\![\sigma\mathcal{B}\ \mathcal{E}]\!]\eta & \stackrel{\Delta}{=} [\![\mathcal{E}]\!]\eta[\sigma f \in \mathbb{B}^I.\ \lambda i{\in}I.[\![\omega_i]\!]([\![\mathcal{E}]\!]\eta[f/X_I])/X_I]. \end{array}$$

The solution of an IBES assigns a value to *every* proposition variable that occurs as a binding variable in the IBES. Often, only the value for specific proposition variables is sought, *e.g.* in local model checking. In such a case, equations that are unimportant to the solution of that variable can be pruned, yielding a smaller IBES, or even a BES. This follows from the following results.

Lemma 2. *Let $\mathcal{F}$ and $\mathcal{G}$ be IBESs. Let η be an arbitrary environment. Then:*

$$\mathsf{occ}(\mathcal{G}) \cap \mathsf{bnd}(\mathcal{F}) = \emptyset \quad \textit{implies} \quad \forall X \notin \mathsf{bnd}(\mathcal{F}) : [\![\mathcal{F}\ \mathcal{G}]\!]\eta(X) = [\![\mathcal{G}]\!]\eta(X).$$

Proof. We use induction on the number of blocks in $\mathcal{F}$.

- Base case: $\mathcal{F}$ consists of zero blocks. This means that $\mathcal{F} = \epsilon$. Let η be an arbitrary environment. Then $[\![\mathcal{F}\ \mathcal{G}]\!]\eta(X) = [\![\epsilon\ \mathcal{G}]\!]\eta(X) = [\![\mathcal{G}]\!]\eta(X)$ for all proposition variables X.
- Inductive step: assume that for some IBES $\mathcal{F}'$ we have for all θ:

  (IH) $\mathsf{occ}(\mathcal{G}) \cap \mathsf{bnd}(\mathcal{F}') = \emptyset$ implies $\forall X \notin \mathsf{bnd}(\mathcal{F}) : [\![\mathcal{F}'\ \mathcal{G}]\!]\theta(X) = [\![\mathcal{G}]\!]\theta(X)$.

  Assume that $\mathsf{occ}(\mathcal{G}) \cap \mathsf{bnd}(\sigma\mathcal{B}\ \mathcal{F}') = \emptyset$. Let $X \notin \mathsf{bnd}(\sigma\mathcal{B}\ \mathcal{F}')$. Then:

  $[\![\sigma\mathcal{B}\ \mathcal{F}'\ \mathcal{G}]\!]\eta(X)$
  $= [\![\mathcal{F}'\ \mathcal{G}]\!]\eta[\sigma X_I.\ \mathcal{B}([\![\mathcal{F}'\ \mathcal{G}]\!]\eta)/X_I](X)$
  $= \{\ \mathsf{occ}(\mathcal{G}) \cap \mathsf{bnd}(\sigma\mathcal{B}\ \mathcal{F}') = \emptyset$ implies $\mathsf{occ}(\mathcal{G}) \cap \mathsf{bnd}(\mathcal{F}') = \emptyset$, apply (IH) $\}$
  $[\![\mathcal{G}]\!]\eta[\sigma X_I.\ \mathcal{B}([\![\mathcal{F}'\ \mathcal{G}]\!]\eta)/X_I](X)$
  $= \{\ X_I \cap \mathsf{occ}(\mathcal{G}) = \emptyset, X \notin X_I\ \}$
  $[\![\mathcal{G}]\!]\eta(X)$. □

Proposition 1. *Let $\mathcal{E}, \mathcal{F}, \mathcal{G}$ be arbitrary IBESs. Then for all environments η:*

$$\mathsf{occ}(\mathcal{E}\ \mathcal{G}) \cap \mathsf{bnd}(\mathcal{F}) = \emptyset \quad \textit{implies} \quad \forall X \notin \mathsf{bnd}(\mathcal{F}) : [\![\mathcal{E}\ \mathcal{F}\ \mathcal{G}]\!]\eta(X) = [\![\mathcal{E}\ \mathcal{G}]\!]\eta(X).$$

Proof. We use induction on the number of blocks in $\mathcal{E}$.

- Base case: $\mathcal{E}$ consists of zero blocks. Then $\mathsf{occ}(\mathcal{G}) \cap \mathsf{bnd}(\mathcal{F}) = \emptyset$ and by Proposition 1 we have $[\![\mathcal{F}\ \mathcal{G}]\!]\eta(X) = [\![\mathcal{G}]\!]\eta(X)$ for all $X \notin \mathsf{bnd}(\mathcal{F})$.
- Inductive step: assume that for an IBES $\mathcal{E}'$ and all environments θ we have:

  (IH) $\mathsf{occ}(\mathcal{E}'\ \mathcal{G}) \cap \mathsf{bnd}(\mathcal{F}) = \emptyset$ implies $\forall X \notin \mathsf{bnd}(\mathcal{F}) : [\![\mathcal{E}'\ \mathcal{F}\ \mathcal{G}]\!]\eta(X) = [\![\mathcal{E}'\ \mathcal{G}]\!]\eta(X)$.

  Suppose that $\mathsf{occ}(\sigma\mathcal{B}\ \mathcal{E}'\ \mathcal{G}) \cap \mathsf{bnd}(\mathcal{F}) = \emptyset$. Let $X \notin \mathsf{bnd}(\mathcal{F})$. Then:

  $[\![\sigma\mathcal{B}\ \mathcal{E}'\ \mathcal{F}\ \mathcal{G}]\!]\eta(X)$
  $= \{$ Definition of semantics $\}$
  $[\![\mathcal{E}'\ \mathcal{F}\ \mathcal{G}]\!]\eta[\sigma X_I.\ \mathcal{B}([\![\mathcal{E}'\ \mathcal{F}\ \mathcal{G}]\!]\eta)/X_I](X)$
  $= \{$ apply (IH)$\}$
  $[\![\mathcal{E}'\ \mathcal{G}]\!]\eta[\sigma X_I.\ \mathcal{B}([\![\mathcal{E}'\ \mathcal{F}\ \mathcal{G}]\!]\eta)/X_I](X)$
  $= \{\ \mathsf{occ}(\sigma\mathcal{B}\ \mathcal{E}'\ \mathcal{G}) \cap \mathsf{bnd}(\mathcal{F}) = \emptyset$ implies $\mathcal{B}([\![\mathcal{E}'\ \mathcal{F}\ \mathcal{G}]\!]\eta) = \mathcal{B}([\![\mathcal{E}'\ \mathcal{G}]\!]\eta)\ \}$
  $[\![\mathcal{E}'\ \mathcal{G}]\!]\eta[\sigma X_I.\ \mathcal{B}([\![\mathcal{E}'\ \mathcal{G}]\!]\eta)/X_I](X)$
  $= \{$ Definition of semantics $\}$
  $[\![\sigma\mathcal{B}\ \mathcal{E}'\ \mathcal{G}]\!]\eta(X)$. □

Algorithm 1 The instantiation algorithm $\mathsf{Inst}_{\mathcal{P}}$

For any $\mathcal{P} \subseteq \mathcal{X}$, with $\mathcal{P} \neq \emptyset$:

$$\mathsf{Inst}_{\emptyset}(\mathcal{E}) \stackrel{\Delta}{=} \mathcal{E}$$
$$\mathsf{Inst}_{\mathcal{P}}(\epsilon) \stackrel{\Delta}{=} \epsilon$$
$$\mathsf{Inst}_{\mathcal{P}}((\sigma X(d{:}D, e{:}E) = \varphi)\ \mathcal{E}) \stackrel{\Delta}{=} \begin{cases} \{(\sigma X_v(e{:}E) = \mathsf{Sub}_{\mathcal{P}}(\varphi[v/d])) \mid v{\in}D\}\ \mathsf{Inst}_{\mathcal{P}}(\mathcal{E}) & \text{if } X{\in}\mathcal{P} \\ (\sigma X(d{:}D, e{:}E) = \mathsf{Sub}_{\mathcal{P}}(\varphi))\ \mathsf{Inst}_{\mathcal{P}}(\mathcal{E}) & \text{otherwise} \end{cases}$$

where

$$\mathsf{Sub}_{\emptyset}(\varphi) \stackrel{\Delta}{=} \varphi$$
$$\mathsf{Sub}_{\mathcal{P}}(b) \stackrel{\Delta}{=} b$$
$$\mathsf{Sub}_{\mathcal{P}}(X(d,e)) \stackrel{\Delta}{=} \begin{cases} \bigvee_{v \in D}(v = d \wedge X_v(e[v/d])) & \text{if } X{\in}\mathcal{P} \\ X(d,e) & \text{otherwise} \end{cases}$$
$$\mathsf{Sub}_{\mathcal{P}}(\varphi_1 \oplus \varphi_2) \stackrel{\Delta}{=} \mathsf{Sub}_{\mathcal{P}}(\varphi_1) \oplus \mathsf{Sub}_{\mathcal{P}}(\varphi_2)$$
$$\mathsf{Sub}_{\mathcal{P}}(\mathsf{Q}d{:}D.\ \varphi) \stackrel{\Delta}{=} \mathsf{Q}d{:}D.\ \mathsf{Sub}_{\mathcal{P}}(\varphi)$$

3 Instantiation on finite domains

In this section, we define a transformation on PBESs that reduces the complexity of selected equations in the PBES. This often allows for solving such PBESs, either by hand or by means of automated tool support. The transformation is given as Algorithm 1. Since the main proof is quite involved, we first prove correctness of this algorithm for $|\mathcal{P}| = 1$, *i.e.* when it instantiates a single equation (Section 3.1). The correctness proof for the general case (arbitrary $\mathcal{P}$) is given in Section 3.2 and relies on the results of Section 3.1.

Without loss of generality, we assume that all predicate variables in this section are either of type $D \times E \to B$ or of type $E \to B$, for some finite sort D and some possibly infinite sort E. The finite sort D is used as the sort that is instantiated for a given predicate variable. We use the sort F when we mean either domain D or $D \times E$. To each predicate variable $X{:}D \times E \to B$, we associate a finite set of predicate variables $\mathsf{all}(X) \stackrel{\Delta}{=} \{X_d{:}E \to B \mid d{\in}D\}$. For any PBES $\mathcal{E}$, we say that the predicate variable X is *instantiation-fresh* for $\mathcal{E}$ iff $\mathsf{all}(X) \cap \mathsf{var}(\mathcal{E}) = \emptyset$.

3.1 Instantiation for a single predicate variable

Instantiation replaces an equation $(\sigma X(d{:}D, e{:}E) = \varphi)$ by an entire PBES $(\sigma X_{d_1}(e{:}E) = \varphi_{d_1}) \cdots (\sigma X_{d_n}(e{:}E) = \varphi_{d_n})$. The transformation can be lifted to general PBESs and to arbitrary subsets of binding variables (see Algorithm 1). Although the basic idea of the transformation is elementary, the devil is in the detail: careful bookkeeping and a naming scheme have to be applied to make the transformation work. This is taken care of by the function $\mathsf{Sub}_{\mathcal{P}}$ that is used

in the main transformation $\mathsf{Inst}_{\mathcal{P}}$. It ensures that new predicate variables are introduced correctly in the right-hand sides of the equations of a PBES. In the definition of $\mathsf{Sub}_{\mathcal{P}}$, the operand $\bigvee_{v \in D}$ abbreviates a finite disjunction over all basic elements v in D.

The soundness of the transformation is far from obvious due to the newly introduced predicate variables. We prove that the transformation indeed preserves the solution of the original PBES, and claim a precise correspondence between the original PBES and the transformed PBES. In order to facilitate the proof of the main claims of this section, we first address several lemmata concerning the functions $\mathsf{Sub}_{\{X\}}$ and $\mathsf{Inst}_{\{X\}}$ to which we refer as Sub_X and Inst_X for conciseness.

Lemma 3. *Let φ be a predicate formula and $X{:}D \times E \to B$ be a predicate variable. Let η be an environment such that $\eta(X)(\llbracket v \rrbracket) = \eta(X_v)$ for all $v \in D$. Then for any environment ε:*

$$\llbracket \varphi \rrbracket \eta \varepsilon = \llbracket \mathsf{Sub}_X(\varphi) \rrbracket \eta \varepsilon.$$

Proof. Let ε be a data environment. We prove the statement by induction on the structure of φ.

1. $\varphi \equiv b$. This holds trivially.
2. $\varphi \equiv X(d, e)$. Then:

$$\begin{aligned}
\llbracket X(d,e) \rrbracket \eta_1 \varepsilon &= \eta(X)(\llbracket d \rrbracket \varepsilon)(\llbracket e \rrbracket \varepsilon) \\
&= \textstyle\bigvee_{v \in D}(\llbracket v \rrbracket = \llbracket d \rrbracket \varepsilon \wedge \eta(X)(\llbracket v \rrbracket)(\llbracket e \rrbracket \varepsilon)) \\
&= \textstyle\bigvee_{v \in D}(\llbracket v \rrbracket = \llbracket d \rrbracket \varepsilon \wedge \eta(X_v)(\llbracket e \rrbracket \varepsilon)) \\
&= \llbracket \textstyle\bigvee_{v \in D}(v = d \wedge X_v(e)) \rrbracket \eta \varepsilon \\
&= \llbracket \mathsf{Sub}_X(X(d,e)) \rrbracket \eta \varepsilon.
\end{aligned}$$

3. $\varphi \equiv Y(d, e)$ for $Y \neq X$. This holds trivially.
4. We assume for formulae φ_i, where $i \in \{1, 2\}$:

   (IH) $\qquad \llbracket \varphi_i \rrbracket \eta \varepsilon = \llbracket \mathsf{Sub}_X(\varphi_i) \rrbracket \eta \varepsilon.$

   (a) $\varphi \equiv \varphi_1 \oplus \varphi_2$. Then:

$$\begin{aligned}
\llbracket \varphi_1 \oplus \varphi_2 \rrbracket \eta \varepsilon &= \llbracket \varphi_1 \rrbracket \eta \varepsilon \oplus \llbracket \varphi_2 \rrbracket \eta \varepsilon \\
&\overset{\text{(IH)}}{=} \llbracket \mathsf{Sub}_X(\varphi_1) \rrbracket \eta \varepsilon \oplus \llbracket \mathsf{Sub}_X(\varphi_2) \rrbracket \eta \varepsilon \\
&= \llbracket \mathsf{Sub}_X(\varphi_1 \oplus \varphi_2) \rrbracket \eta \varepsilon.
\end{aligned}$$

   (b) $\varphi \equiv \mathsf{Q} f{:}F.\ \varphi_1$. Then:

$$\begin{aligned}
\llbracket \mathsf{Q} f{:}F.\ \varphi_1 \rrbracket \eta \varepsilon &= \mathsf{Q} w \in \mathbb{F} : \llbracket \varphi_1 \rrbracket \eta(\varepsilon[w/f]) \\
&\overset{\text{(IH)}}{=} \mathsf{Q} w \in \mathbb{F} : \llbracket \mathsf{Sub}_X(\varphi_1) \rrbracket \eta(\varepsilon[w/f]) \\
&= \llbracket \mathsf{Sub}_X(\mathsf{Q} f{:}F.\ \varphi_1) \rrbracket \eta \varepsilon.
\end{aligned}$$

□

Lemma 4. *Let $X{:}D \times E \to B$ be a predicate variable. Let $\mathcal{E}$ be a PBES for which X is instantiation-fresh and $X \notin \mathsf{bnd}(\mathcal{E})$. Let η be an environment such that $\eta(X)(\llbracket v \rrbracket) = \eta(X_v)$ for all $v \in D$. Then for arbitrary environment ε, we have:*

$$\llbracket \mathcal{E} \rrbracket \eta\varepsilon = \llbracket \mathsf{Inst}_X(\mathcal{E}) \rrbracket \eta\varepsilon.$$

Proof. Let ε be an arbitrary data environment, and let η be a predicate environment satisfying the conditions of the lemma. We prove the lemma by induction on the length of $\mathcal{E}$. If $\mathcal{E}$ is of length 0 then: $\llbracket \epsilon \rrbracket \eta\varepsilon = \eta = \llbracket \mathsf{Inst}_X(\epsilon) \rrbracket \eta\varepsilon$. Suppose $\mathcal{E}$ is of length $m+1$, and for all $\mathcal{E}'$ of length m, we have, for all environments υ:

$$\text{(IH)} \qquad \llbracket \mathcal{E}' \rrbracket \eta\upsilon = \llbracket \mathsf{Inst}_X(\mathcal{E}') \rrbracket \eta\upsilon.$$

Necessarily, $\mathcal{E}$ is of the form $(\sigma Z(f{:}F) = \varphi)\ \mathcal{F}$, where $\mathcal{F}$ is of length m. We derive:

$$\begin{aligned}
&\llbracket \mathcal{E} \rrbracket \eta\varepsilon \\
&= \llbracket (\sigma Z(f{:}F) = \varphi)\ \mathcal{F} \rrbracket \eta\varepsilon \\
&= \llbracket \mathcal{F} \rrbracket \eta[(\sigma g \in \mathbb{B}^{\mathbb{F}}.\ \lambda v{\in}\mathbb{F}.\ \llbracket \varphi \rrbracket (\llbracket \mathcal{F} \rrbracket \eta[g/Z]\varepsilon)\varepsilon[v/f])/Z]\varepsilon \\
&\overset{*}{=} \llbracket \mathcal{F} \rrbracket \eta[(\sigma g \in \mathbb{B}^{\mathbb{F}}.\ \lambda v{\in}\mathbb{F}.\ \llbracket \mathsf{Sub}_X(\varphi) \rrbracket (\llbracket \mathcal{F} \rrbracket \eta[g/Z]\varepsilon)\varepsilon[v/f])/Z]\varepsilon \\
&\overset{\text{(IH)}}{=} \llbracket \mathsf{Inst}_X(\mathcal{F}) \rrbracket \eta[(\sigma g \in \mathbb{B}^{\mathbb{F}}.\ \lambda v{\in}\mathbb{F}.\ \llbracket \mathsf{Sub}_X(\varphi) \rrbracket (\llbracket \mathsf{Inst}_X(\mathcal{F}) \rrbracket \eta[g/Z]\varepsilon)\varepsilon[v/f])/Z]\varepsilon \\
&= \llbracket \mathsf{Inst}_X((\sigma Z(f{:}F) = \varphi)\ \mathcal{F}) \rrbracket \eta\varepsilon \\
&= \llbracket \mathsf{Inst}_X(\mathcal{E}) \rrbracket \eta\varepsilon.
\end{aligned}$$

At * we used the following equivalence:

$$(\sigma g \in \mathbb{B}^{\mathbb{F}}.\llbracket \varphi \rrbracket (\llbracket \mathcal{F} \rrbracket \eta[g/Z]\varepsilon)\varepsilon) = (\sigma g \in \mathbb{B}^{\mathbb{F}}.\llbracket \mathsf{Sub}_X(\varphi) \rrbracket (\llbracket \mathcal{F} \rrbracket \eta[g/Z]\varepsilon)\varepsilon)$$

which follows readily from Lemma 3. Observe that this lemma applies because $(\llbracket \mathcal{F} \rrbracket \eta[g/Z]\varepsilon)(X)(\llbracket v \rrbracket) = (\llbracket \mathcal{F} \rrbracket \eta[g/Z]\varepsilon)(X_v)$ for all $v \in D$ by assumption on η, instantiation-freshness of X in $\mathcal{E}$, $X \notin \mathsf{bnd}(\mathcal{F})$ and Lemma 1. □

Suppose we have a PBES $\mathcal{E}$ in which the first equation is for variable $X{:}D \times E \to B$ and the domain D of X is instantiated in that PBES, *i.e.* we use the transformation $\mathsf{Inst}_X(\mathcal{E})$. The PBES resulting from the transformation will consist of $|D|$ equations replacing the single equation for X, plus the remaining $|\mathcal{E}|-1$ equations from $\mathcal{E}$. The following lemma states that the solution to X in the original PBES and the solutions to its instantiated counterparts in the resulting PBES correspond. Note that it does not state that the transformation does not have undesirable side-effects. This latter problem is addressed in Lemma 6 (below the following lemma) which states that non-instantiated variables are unaffected.

Lemma 5. *Let $\mathcal{F}$ be a PBES of the form $(\sigma X(d{:}D, e{:}E) = \varphi)\ \mathcal{E}$ such that X is instantiation-fresh for $\mathcal{F}$. Then for any environment η, ε:*

$$\forall v \in D : (\llbracket \mathsf{Inst}_X(\mathcal{F}) \rrbracket \eta\varepsilon)(X_v) = ((\llbracket \mathcal{F} \rrbracket \eta\varepsilon)(X))(\llbracket v \rrbracket).$$

Proof. Assume that $D = \{v_1, \ldots, v_n\}$; then $|D| = |\mathbb{D}| = n$ and take $1 \leq i \leq n$. We abbreviate $\mathsf{Inst}_X(\mathcal{E})$ by $\mathcal{E}_\iota$. First, we rewrite the left-hand side of the equality as follows:

$$\begin{aligned}
&(\llbracket \mathsf{Inst}_X(\mathcal{F}) \rrbracket \eta\varepsilon)(X_{v_i}) \\
=^\dagger\ &\pi_i\Big(\big(\sigma(g_{v_1}, \ldots, g_{v_n}) \in \mathbb{B}^{\mathbb{E}^n}. \\
&(\lambda w \in \mathbb{E}.\ \llbracket \mathsf{Sub}_X(\varphi[v_1/d]) \rrbracket (\llbracket \mathcal{E}_\iota \rrbracket \eta[(g_{v_1}, \ldots, g_{v_n})/(X_{v_1}, \ldots, X_{v_n})]\varepsilon)\varepsilon[w/e], \ldots, \\
&\lambda w \in \mathbb{E}.\ \llbracket \mathsf{Sub}_X(\varphi[v_n/d]) \rrbracket (\llbracket \mathcal{E}_\iota \rrbracket \eta[(g_{v_1}, \ldots, g_{v_n})/(X_{v_1}, \ldots, X_{v_n})]\varepsilon)\varepsilon[w/e])\big)\Big) \\
=^\ddagger\ &\big(\sigma g \in \mathbb{B}^{\mathbb{D}\to\mathbb{E}}.\ (\lambda u \in \mathbb{D}. \lambda w \in \mathbb{E}. \\
&\quad \llbracket \mathsf{Sub}_X(\varphi) \rrbracket (\llbracket \mathcal{E}_\iota \rrbracket \eta[g(\llbracket v_1 \rrbracket)/X_{v_1}, \ldots, g(\llbracket v_n \rrbracket)/X_{v_n}]\varepsilon)\varepsilon[u/d, w/e])\big)(\llbracket v_i \rrbracket).
\end{aligned}$$

At † we used Bekič's theorem [1] to replace n nested σ-fixpoints by one simultaneous σ-fixpoint over an n-tuple, and the fact that $X_{v_i} \in \mathsf{bnd}(\mathsf{Inst}_X(\mathcal{F}))$. At ‡ we used the assumption that the data theory is fully abstract, and the isomorphism between $\mathbb{B}^{\mathbb{E}^{|\mathbb{D}|}}$ and $\mathbb{B}^{\mathbb{D}\to\mathbb{E}}$ to replace a tuple of functions $(g_{v_1}, \ldots, g_{v_n}){:}(\mathbb{E} \to \mathbb{B})^n$ by a single function $g{:}\mathbb{D} \to \mathbb{E} \to \mathbb{B}$ such that for any data element $u \in D$: $g(\llbracket u \rrbracket) = g_u$.

For the right-hand side, we calculate:

$$\begin{aligned}
&(\llbracket \mathcal{F} \rrbracket \eta\varepsilon)(X(\llbracket v_i \rrbracket\varepsilon)) \\
&= \big(\sigma f \in \mathbb{B}^{\mathbb{D}\times\mathbb{E}}.\ \lambda(u, w) \in (\mathbb{D} \times \mathbb{E}).\ \llbracket \varphi \rrbracket (\llbracket \mathcal{E} \rrbracket \eta[f/X]\varepsilon)\varepsilon[(u, w)/(d, e)]\big)(\llbracket v_i \rrbracket) \\
&= \big(\sigma f \in \mathbb{B}^{\mathbb{D}\to\mathbb{E}}.\ \lambda u \in \mathbb{D}.\ \lambda w \in \mathbb{E}.\ \llbracket \varphi \rrbracket (\llbracket \mathcal{E} \rrbracket \eta[f/X]\varepsilon)\varepsilon[u/d, w/e]\big)(\llbracket v_i \rrbracket).
\end{aligned}$$

So it suffices to show the following equivalence:

$$\begin{aligned}
&\big(\sigma f \in \mathbb{B}^{\mathbb{D}\to\mathbb{E}}.\ \lambda u \in \mathbb{D}.\ \lambda w \in \mathbb{E}.\ \llbracket \varphi \rrbracket (\llbracket \mathcal{E} \rrbracket \eta[f/X]\varepsilon)\varepsilon[u/d, w/e]\big) \\
&= \big(\sigma g \in \mathbb{B}^{\mathbb{D}\to\mathbb{E}}.\ (\lambda u \in \mathbb{D}.\ \lambda w \in \mathbb{E}. \\
&\qquad \llbracket \mathsf{Sub}_X(\varphi) \rrbracket (\llbracket \mathcal{E}_\iota \rrbracket \eta[g(\llbracket v_1 \rrbracket)/X_{v_1}, \ldots, g(\llbracket v_n \rrbracket)/X_{v_n}]\varepsilon)\varepsilon[u/d, w/e])\big)
\end{aligned}$$

which follows readily from:

$$\begin{aligned}
(*) \qquad &\llbracket \varphi \rrbracket (\llbracket \mathcal{E} \rrbracket \eta[h/X]\varepsilon)\upsilon \\
&= \llbracket \mathsf{Sub}_X(\varphi) \rrbracket (\llbracket \mathcal{E}_\iota \rrbracket \eta[h(\llbracket v_1 \rrbracket)/X_{v_1}, \ldots, h(\llbracket v_n \rrbracket)/X_{v_n}]\varepsilon)\upsilon
\end{aligned}$$

for all environments υ and $h \in \mathbb{B}^{\mathbb{D}\to\mathbb{E}}$. Let υ be an environment and $h \in \mathbb{B}^{\mathbb{D}\to\mathbb{E}}$. We prove (*) using Lemmata 3 and 4 as follows:

$$\begin{aligned}
&\llbracket \varphi \rrbracket (\llbracket \mathcal{E} \rrbracket \eta[h/X]\varepsilon)\upsilon \\
=^\dagger\ &\llbracket \varphi \rrbracket (\llbracket \mathcal{E} \rrbracket \eta[h/X][h(\llbracket v_1 \rrbracket)/X_{v_1}, \ldots, h(\llbracket v_n \rrbracket)/X_{v_n}]\varepsilon)\upsilon \\
=\ &\llbracket \mathsf{Sub}_X(\varphi) \rrbracket (\llbracket \mathcal{E} \rrbracket \eta[h/X][h(\llbracket v_1 \rrbracket)/X_{v_1}, \ldots, h(\llbracket v_n \rrbracket)/X_{v_n}]\varepsilon)\upsilon \\
=\ &\llbracket \mathsf{Sub}_X(\varphi) \rrbracket (\llbracket \mathcal{E}_\iota \rrbracket \eta[h/X][h(\llbracket v_1 \rrbracket)/X_{v_1}, \ldots, h(\llbracket v_n \rrbracket)/X_{v_n}]\varepsilon)\upsilon \\
=^\ddagger\ &\llbracket \mathsf{Sub}_X(\varphi) \rrbracket (\llbracket \mathcal{E}_\iota \rrbracket \eta[h(\llbracket v_1 \rrbracket)/X_{v_1}, \ldots, h(\llbracket v_n \rrbracket)/X_{v_n}]\varepsilon)\upsilon.
\end{aligned}$$

At ‡ we used $X \notin \mathsf{occ}(\mathsf{Sub}_X(\varphi))$ and at † we used instantiation-freshness of X in equation system $\mathcal{F}$. □

Lemma 6. *Let $\mathcal{F} \stackrel{\Delta}{=} (\sigma X(d{:}D, e{:}E) = \varphi)\ \mathcal{E}$ be a PBES and let X be instantiation-fresh for $\mathcal{F}$. Then for all environments η, ε:*

$$\forall Y \in \mathcal{X} : Y \notin \mathsf{all}(X) \cup \{X\} \implies (\llbracket \mathsf{Inst}_X(\mathcal{F}) \rrbracket \eta\varepsilon)(Y) = (\llbracket \mathcal{F} \rrbracket \eta\varepsilon)(Y).$$

Proof. Let $Y \in \mathcal{X}$ such that $Y \notin \mathsf{all}(X) \cup \{X\}$. Let $g : \mathbb{D} \times \mathbb{E} \to \mathbb{B}$ be such that:

$$\forall v \in D : g(\llbracket v \rrbracket) = (\llbracket \mathsf{Inst}_X(\mathcal{F}) \rrbracket \eta\varepsilon)(X_v).$$

Then by Lemma 5, we have $g = (\llbracket \mathcal{F} \rrbracket \eta\varepsilon)(X)$ and:

$$\begin{array}{ll} & (\llbracket \mathsf{Inst}_X(\mathcal{F}) \rrbracket \eta\varepsilon)(Y) \\ = & (\llbracket \mathsf{Inst}_X(\mathcal{E}) \rrbracket \eta[g(\llbracket v_1 \rrbracket)/X_{v_1}, \ldots, g(\llbracket v_n \rrbracket)/X_{v_n}]\varepsilon)(Y) \\ = & (\llbracket \mathsf{Inst}_X(\mathcal{E}) \rrbracket \eta[g/X][g(\llbracket v_1 \rrbracket)/X_{v_1}, \ldots, g(\llbracket v_n \rrbracket)/X_{v_n}]\varepsilon)(Y) \\ =^{\dagger} & (\llbracket \mathcal{E} \rrbracket \eta[g/X][g(\llbracket v_1 \rrbracket)/X_{v_1}, \ldots, g(\llbracket v_n \rrbracket)/X_{v_n}]\varepsilon)(Y) \\ = & (\llbracket \mathcal{E} \rrbracket \eta[g/X]\varepsilon)(Y) \\ = & (\llbracket \mathcal{F} \rrbracket \eta\varepsilon)(Y) \end{array}$$

where at † we used Lemma 4. □

We now prove our main claim of this section, stating that instantiation of a single predicate variable is sound for arbitrary PBESs, *i.e.* not only for those equation systems for which the first equation is instantiated, but also equation systems in which the equation for the instantiated variable occurs at some arbitrary position in the equation system.

Proposition 2. *Let $\mathcal{E}$ be a PBES and $X \in \mathsf{bnd}(\mathcal{E})$ instantiation-fresh for $\mathcal{E}$. Then for all environments η, ε:*

(2a) $\forall v \in D : (\llbracket \mathsf{Inst}_X(\mathcal{E}) \rrbracket \eta\varepsilon)(X_v) = (\llbracket \mathcal{E} \rrbracket \eta\varepsilon)(X(\llbracket v \rrbracket))$

(2b) $\forall Y \in \mathcal{X} : Y \notin \mathsf{all}(X) \cup \{X\} \implies (\llbracket \mathsf{Inst}_X(\mathcal{E}) \rrbracket \eta\varepsilon)(Y) = (\llbracket \mathcal{E} \rrbracket \eta\varepsilon)(Y).$

Proof. Observe that $\mathcal{E}$ is of the following form:

$$\mathcal{E} \stackrel{\Delta}{=} \mathcal{E}_1\ \mathcal{F} \quad \text{with} \quad \mathcal{F} \stackrel{\Delta}{=} (\sigma X(d{:}D, e{:}E) = \varphi)\ \mathcal{E}_2$$

for some predicate formula φ and PBESs $\mathcal{E}_1$ and $\mathcal{E}_2$. We prove the claim by induction on the structure of $\mathcal{E}_1$. For $\mathcal{E}_1 = \epsilon$, (2a) and (2b) follow immediately due to Lemmata 5 and 6, respectively. Suppose $\mathcal{E}_1 = (\sigma' Z(f{:}F) = \psi)\ \mathcal{E}'$ for some PBES $\mathcal{E}'$. We assume as induction hypotheses, for all environments η', ε':

(IHa)
$\forall v \in D : (\llbracket \mathsf{Inst}_X(\mathcal{E}'\ \mathcal{F}) \rrbracket \eta'\varepsilon')(X_v) = (\llbracket \mathcal{E}'\ \mathcal{F} \rrbracket \eta'\varepsilon')(X(\llbracket v \rrbracket))$

(IHb)
$\forall Y \in \mathcal{X} : Y \notin \mathsf{all}(X) \cup \{X\} \implies (\llbracket \mathsf{Inst}_X(\mathcal{E}'\ \mathcal{F}) \rrbracket \eta'\varepsilon')(Y) = (\llbracket \mathcal{E}'\ \mathcal{F} \rrbracket \eta'\varepsilon')(Y).$

Let $v \in D$. Then for (2a) we derive:

$$\begin{array}{ll}
& (\llbracket \mathsf{Inst}_X((\sigma' Z(f{:}F) = \psi)\ \mathcal{E}'\ \mathcal{F})\rrbracket \eta\varepsilon)(X_v) \\
= & (\llbracket (\sigma' Z(f{:}F) = \mathsf{Sub}_X(\psi))\ \mathsf{Inst}_X(\mathcal{E}'\ \mathcal{F})\rrbracket \eta\varepsilon)(X_v) \\
= & (\llbracket \mathsf{Inst}_X(\mathcal{E}'\ \mathcal{F})\rrbracket \eta[(\sigma' h{\in}\mathbb{B}^{\mathbb{F}}. \\
& \qquad \lambda u \in \mathbb{F}.\ \llbracket \mathsf{Sub}_X(\psi)\rrbracket(\llbracket \mathsf{Inst}_X(\mathcal{E}'\ \mathcal{F})\rrbracket \eta[h/Z]\varepsilon))/Z]\varepsilon[u/f])(X_v) \\
=^{\dagger} & (\llbracket \mathcal{E}'\ \mathcal{F}\rrbracket \eta[(\sigma' h{\in}\mathbb{B}^{\mathbb{F}}.\ \lambda u \in \mathbb{F}.\ \llbracket \psi\rrbracket(\llbracket \mathcal{E}'\ \mathcal{F}\rrbracket \eta[h/Z]\varepsilon)\varepsilon[u/f]/Z]\varepsilon)(X(\llbracket v\rrbracket)) \\
= & (\llbracket (\sigma' Z(f{:}F) = \psi)\ \mathcal{E}'\ \mathcal{F}\rrbracket \eta\varepsilon)(X(\llbracket v\rrbracket)).
\end{array}$$

At † we used the induction hypothesis (IHa) and Lemma 3 using both (IHa) and (IHb). The derivation for (2b) follows the same line of reasoning and is therefore omitted. □

Example 1. Consider the following PBES $\mathcal{E}$

$$(\nu X(b{:}B) = \exists n{:}N.\ Y(n) \wedge b)(\mu Y(n{:}N) = X(n \geq 10))$$

Instantiation of X in $\mathcal{E}$ yields the PBES $\mathcal{E}'$ below (after minor rewriting):

$$(\nu X_{\top} = \exists n{:}N.\ Y(n))\ (\nu X_{\bot} = \bot)\ (\mu Y(n{:}N) = (n \geq 10 \wedge X_{\top}) \vee (n < 10 \wedge X_{\bot}))$$

Using the syntactic operations of migration and backward substitution [7], and subsequent logic rewriting, the PBES $\mathcal{E}'$ can be solved, yielding:

$$(\nu X_{\top} = \top)\ (\mu Y(n{:}N) = n \geq 10)\ (\nu X_{\bot} = \bot)$$

We have, for arbitrary environment η, ε the following correspondences:

- $(\llbracket \mathcal{E}\rrbracket \eta\varepsilon)(X)(\top) = (\llbracket \mathcal{E}'\rrbracket \eta\varepsilon(X_{\top}) = \top$,
- $(\llbracket \mathcal{E}\rrbracket \eta\varepsilon)(X)(\bot) = (\llbracket \mathcal{E}'\rrbracket \eta\varepsilon(X_{\bot}) = \bot$,
- $(\llbracket \mathcal{E}\rrbracket \eta\varepsilon)(Y) = (\llbracket \mathcal{E}'\rrbracket \eta\varepsilon(Y) = (\lambda n{:}\mathbb{N}.\ n \geq 10)$.

Instantiation allows for solving a rather complex PBES $\mathcal{E}$ using standard PBES manipulation techniques and instantiation of a single predicate variable. □

3.2 Simultaneous Instantiation

Instantiation for a set of variables $\mathcal{P}$ in a PBES can be achieved by applying Inst_X for all $X \in \mathcal{P}$ repeatedly. However, sound as this strategy may be, it is undesirable as it is highly inefficient. We therefore prove that the instantiation algorithm $\mathsf{Inst}_{\mathcal{P}}$ is sound for a general set $\mathcal{P}$. We do so by showing that applying $\mathsf{Inst}_{\mathcal{P}}$ yields an equation system that is syntactically equivalent to successively applying Inst_X for every $X \in \mathcal{P}$. First, we prove several lemmata concerning $\mathsf{Inst}_{\mathcal{P}}$ and $\mathsf{Sub}_{\mathcal{P}}$.

Lemma 7. *Let φ be a predicate formula and $\mathcal{P}$ be a nonempty set of predicate variables such that for all $X, Y \in \mathcal{P}$, $X \notin \mathsf{all}(Y)$. Then for all $X \in \mathcal{P}$:*

$$\mathsf{Sub}_{\mathcal{P}}(\varphi) = \mathsf{Sub}_X(\mathsf{Sub}_{\mathcal{P}\setminus\{X\}}(\varphi)).$$

Proof. We prove the lemma by structural induction on φ. Let $X \in \mathcal{P}$. Then:

1. $\varphi \equiv b$. Trivial.
2. $\varphi \equiv X(d, e)$. Then:

$$\begin{aligned}\mathsf{Sub}_{\mathcal{P}}(X(d,e)) &= \bigvee_{v \in D}(d = v \wedge X_v(e[v/d])) \\ &= \mathsf{Sub}_X(X(d,e)) \\ &= \mathsf{Sub}_X(\mathsf{Sub}_{\mathcal{P}\setminus\{X\}}(X(d,e)))\end{aligned}$$

3. $\varphi \equiv Y(d, e)$ for some $Y \in \mathcal{X}$ with $Y \neq X$. If $Y \notin \mathcal{P}$ then this case is trivial. If $Y \in \mathcal{P}$ then:

$$\begin{aligned}\mathsf{Sub}_{\mathcal{P}}(Y(d,e)) &= \bigvee_{v \in D}(d = v \wedge Y_v(e[v/d])) \\ &=^{\dagger} \mathsf{Sub}_X(Y_v(e[v/d])) \\ &=^{\ddagger} \mathsf{Sub}_X(\mathsf{Sub}_{\mathcal{P}\setminus\{X\}}(Y(d,e)))\end{aligned}$$

   where at $\dagger$ we used $X \notin \mathsf{all}(Y)$ and at $\ddagger$ we used $Y \in \mathcal{P} \setminus \{X\}$.
4. Assume for predicate formulae φ_i, $i \in \{1, 2\}$:

   (IH) $$\mathsf{Sub}_{\mathcal{P}}(\varphi_i) = \mathsf{Sub}_X(\mathsf{Sub}_{\mathcal{P}\setminus\{X\}}(\varphi_i))$$

   (a) $\varphi \equiv \varphi_1 \oplus \varphi_2$ for some $\oplus \in \{\vee, \wedge\}$. Then:

$$\begin{aligned}\mathsf{Sub}_{\mathcal{P}}(\varphi_1 \oplus \varphi_2) &= \mathsf{Sub}_{\mathcal{P}}(\varphi_1) \oplus \mathsf{Sub}_{\mathcal{P}}(\varphi_2) \\ &=^{\text{IH}} \mathsf{Sub}_X(\mathsf{Sub}_{\mathcal{P}\setminus\{X\}}(\varphi_1)) \oplus \mathsf{Sub}_X(\mathsf{Sub}_{\mathcal{P}\setminus\{X\}}(\varphi_2)) \\ &= \mathsf{Sub}_X(\mathsf{Sub}_{\mathcal{P}\setminus\{X\}}(\varphi_1 \oplus \varphi_2))\end{aligned}$$

   (b) $\varphi \equiv \mathsf{Q}d{:}D.\varphi_1$ for some $\mathsf{Q} \in \{\exists, \forall\}$. Then:

$$\begin{aligned}\mathsf{Sub}_{\mathcal{P}}(\mathsf{Q}d{:}D.\varphi_1) &= \mathsf{Q}d{:}D.\mathsf{Sub}_{\mathcal{P}}(\varphi_1) \\ &=^{\text{IH}} \mathsf{Q}d{:}D.\mathsf{Sub}_X(\mathsf{Sub}_{\mathcal{P}\setminus\{X\}}(\varphi_1)) \\ &= \mathsf{Sub}_X(\mathsf{Sub}_{\mathcal{P}\setminus\{X\}}(\mathsf{Q}d{:}D.\varphi_1))\end{aligned}$$

□

Lemma 8. *Let $\mathcal{P}$ be a nonempty set of predicate variables and $\mathcal{E}$ be a PBES such that $\mathcal{P} \subseteq \mathsf{bnd}(\mathcal{E})$ and the set of predicate variables $\mathcal{P}$ is instantiation-fresh in $\mathcal{E}$. Then for all $X \in \mathcal{P}$:*

$$\mathsf{Inst}_{\mathcal{P}}(\mathcal{E}) = \mathsf{Inst}_X(\mathsf{Inst}_{\mathcal{P}\setminus\{X\}}(\mathcal{E}))$$

Proof. In this proof we rely on Lemma 7 which applies because from $\mathcal{P} \subseteq \mathsf{bnd}(\mathcal{E})$ and $\mathcal{P}$ is instantiation-fresh for $\mathcal{E}$ it follows that for all $X, Y \in \mathcal{P}$, $X \notin \mathsf{all}(Y)$. We prove the lemma by induction on the length of $\mathcal{E}$. The case $\mathcal{E} = \epsilon$ is trivial. Suppose $\mathcal{E} = (\sigma Y(d{:}D, e{:}E) = \varphi)\ \mathcal{E}'$ for some PBES $\mathcal{E}'$. Assume that for all $X \in \mathcal{P}$:

(IH) $$\mathsf{Inst}_{\mathcal{P}}(\mathcal{E}') = \mathsf{Inst}_X(\mathsf{Inst}_{\mathcal{P}\setminus\{X\}}(\mathcal{E}'))$$

Let $X \in \mathcal{P}$. We distinguish three cases:

1. $Y \notin \mathcal{P}$. Then:

$$\begin{aligned}
&\mathsf{Inst}_{\mathcal{P}}((\sigma Y(d{:}D, e{:}E) = \varphi)\ \mathcal{E}') \\
&= \ (\sigma Y(d{:}D, e{:}E) = \mathsf{Sub}_{\mathcal{P}}(\varphi))\ \mathsf{Inst}_{\mathcal{P}}(\mathcal{E}') \\
&=^{\dagger}\ (\sigma Y(d{:}D, e{:}E) = \mathsf{Sub}_{X}(\mathsf{Sub}_{\mathcal{P}\setminus\{X\}}(\varphi)))\ \mathsf{Inst}_{X}(\mathsf{Inst}_{\mathcal{P}\setminus\{X\}}(\mathcal{E}')) \\
&= \ \mathsf{Inst}_{X}((\sigma Y(d{:}D, e{:}E) = \mathsf{Sub}_{\mathcal{P}\setminus\{X\}}(\varphi))\ \mathsf{Inst}_{\mathcal{P}\setminus\{X\}}(\mathcal{E}')) \\
&= \ \mathsf{Inst}_{X}(\mathsf{Inst}_{\mathcal{P}\setminus\{X\}}((\sigma Y(d{:}D, e{:}E) = \varphi)\ \mathcal{E}'))
\end{aligned}$$

2. $Y \in \mathcal{P} \wedge Y \neq X$. Observe that $X \notin \mathsf{all}(Y)$. Then:

$$\begin{aligned}
&\mathsf{Inst}_{\mathcal{P}}((\sigma Y(d{:}D, e{:}E) = \varphi)\ \mathcal{E}') \\
&= \ \{(\sigma Y_v(e{:}E) = \mathsf{Sub}_{\mathcal{P}}(\varphi[v/d])) \mid v{\in}D\}\ \mathsf{Inst}_{\mathcal{P}}(\mathcal{E}') \\
&=^{\dagger}\ \{(\sigma Y_v(e{:}E) = \mathsf{Sub}_{X}(\mathsf{Sub}_{\mathcal{P}\setminus\{X\}}(\varphi[v/d]))) \mid v{\in}D\}\ \mathsf{Inst}_{X}(\mathsf{Inst}_{\mathcal{P}\setminus\{X\}}(\mathcal{E}')) \\
&= \ \mathsf{Inst}_{X}(\{(\sigma Y_v(e{:}E) = \mathsf{Sub}_{\mathcal{P}\setminus\{X\}}(\varphi[v/d])) \mid v{\in}D\}\ \mathsf{Inst}_{\mathcal{P}\setminus\{X\}}(\mathcal{E}')) \\
&= \ \mathsf{Inst}_{X}(\mathsf{Inst}_{\mathcal{P}\setminus\{X\}}((\sigma Y(d{:}D, e{:}E) = \varphi)\ \mathcal{E}'))
\end{aligned}$$

3. $Y = X$. Then:

$$\begin{aligned}
&\mathsf{Inst}_{\mathcal{P}}((\sigma X(d{:}D, e{:}E) = \varphi)\ \mathcal{E}') \\
&= \ \{(\sigma X_v(e{:}E) = \mathsf{Sub}_{\mathcal{P}}(\varphi[v/d])) \mid v{\in}D\}\ \mathsf{Inst}_{\mathcal{P}}(\mathcal{E}') \\
&=^{\dagger}\ \{(\sigma X_v(e{:}E) = \mathsf{Sub}_{X}(\mathsf{Sub}_{\mathcal{P}\setminus\{X\}}(\varphi[v/d]))) \mid v{\in}D\}\ \mathsf{Inst}_{X}(\mathsf{Inst}_{\mathcal{P}\setminus\{X\}}(\mathcal{E}')) \\
&= \ \mathsf{Inst}_{X}((\sigma X(d{:}D, e{:}E) = \mathsf{Sub}_{\mathcal{P}\setminus\{X\}}(\varphi))\ \mathsf{Inst}_{\mathcal{P}\setminus\{X\}}(\mathcal{E}')) \\
&= \ \mathsf{Inst}_{X}(\mathsf{Inst}_{\mathcal{P}\setminus\{X\}}((\sigma X(d{:}D, e{:}E) = \varphi)\ \mathcal{E}'))
\end{aligned}$$

At every † we used Lemma 7 and (IH). □

We introduce the following shorthand notation for functional composition of Inst functions over a set of variables $\mathcal{P}$:

$$\textstyle\bigcirc_{X\in\mathcal{P}}\mathsf{Inst}_X = \begin{cases} \mathcal{I} & \text{if } \mathcal{P} = \emptyset \\ \mathsf{Inst}_Y \circ \bigcirc_{X\in\mathcal{P}\setminus\{Y\}}\mathsf{Inst}_X & \text{for some } Y \in \mathcal{P}, \text{ otherwise} \end{cases}$$

where $\mathcal{I}$ denotes the identity function for PBESs, *i.e.* $\mathcal{I}(\mathcal{E}) = \mathcal{E}$ for all $\mathcal{E}$. We now prove that instantiating a set of variables $\mathcal{P}$ yields the same equation system as successively instantiating for every variable in $\mathcal{P}$ in arbitrary order.

Lemma 9. *Let $\mathcal{P}$ be a set of predicate variables and $\mathcal{E}$ be a PBES such that $\mathcal{P} \subseteq \mathsf{bnd}(\mathcal{E})$ and every predicate variable in $\mathcal{P}$ is instantiation-fresh in $\mathcal{E}$. Then:*

$$\mathsf{Inst}_{\mathcal{P}}(\mathcal{E}) = (\textstyle\bigcirc_{X\in\mathcal{P}}\mathsf{Inst}_X)(\mathcal{E})$$

Proof. The proof goes by induction on the size of $\mathcal{P}$. If $\mathcal{P} = \emptyset$ then trivially: $\mathsf{Inst}_{\mathcal{P}}(\mathcal{E}) = \mathcal{E} = \mathcal{I}(\mathcal{E}) = (\bigcirc_{X\in\mathcal{P}}\mathsf{Inst}_X)(\mathcal{E})$. Otherwise, let $Y \in \mathcal{P}$ and assume:

$$\text{(IH)} \qquad \mathsf{Inst}_{\mathcal{P}\setminus\{Y\}}(\mathcal{E}) = (\textstyle\bigcirc_{X\in\mathcal{P}\setminus\{Y\}}\mathsf{Inst}_X)(\mathcal{E})$$

Then:

$$\begin{aligned}
\mathsf{Inst}_{\mathcal{P}}(\mathcal{E}) &=^{\dagger}\ \mathsf{Inst}_Y(\mathsf{Inst}_{\mathcal{P}\setminus\{Y\}}(\mathcal{E})) \\
&=^{\text{IH}}\ (\mathsf{Inst}_Y \circ \textstyle\bigcirc_{X\in\mathcal{P}\setminus\{Y\}}\mathsf{Inst}_X)(\mathcal{E}) \\
&=\ (\textstyle\bigcirc_{X\in\mathcal{P}}\mathsf{Inst}_P)(\mathcal{E})
\end{aligned}$$

where at † we used Lemma 8. □

Theorem 1. *Let $\mathcal{E}$ be a PBES and $\mathcal{P} \subseteq \mathsf{bnd}(\mathcal{E})$ be a set of predicate variables that is instantiation-fresh in $\mathcal{E}$. Then for all environments η, ε:*

$$\text{(2a)} \quad \forall X{\in}\mathcal{P} : \ \forall v{\in}D : \ (\llbracket \mathsf{Inst}_{\mathcal{P}}(\mathcal{E}) \rrbracket \eta\varepsilon)(X_v) = (\llbracket \mathcal{E} \rrbracket \eta\varepsilon)(X(\llbracket v \rrbracket))$$

$$\text{(2b)} \quad \forall Y{\in}\mathcal{X} : \ Y \notin \mathsf{all}(\mathcal{P}) \cup \mathcal{P} \implies (\llbracket \mathsf{Inst}_{\mathcal{P}}(\mathcal{E}) \rrbracket \eta\varepsilon)(Y) = (\llbracket \mathcal{E} \rrbracket \eta\varepsilon)(Y).$$

Proof. The proof is done by an induction on the size of $\mathcal{P}$, relying on Lemma 9 and Proposition 2 for proving the correctness of instantiating a single predicate variable. □

The above result allows for a full instantiation of a PBES to a BES. This is a sound strategy when (1) all data sorts that occur in the PBES are finite, (2) the PBES is closed and data-closed, and (3) it is possible to rewrite every data term that occurs in the right-hand side expressions of the PBES to either $\top$ or $\bot$. We assume that the latter can be achieved by a data term evaluator eval, which can be implemented using e.g. rewriting technology; the data term evaluator can be lifted to PBESs in a straightforward manner. The following is then a corollary to Lemma 9.

Corollary 1. *Let $\mathcal{E}$ be a PBES. If $\mathcal{E}$ is closed and data-closed and all data sorts occurring in $\mathcal{E}$ are finite and a suitable term rewriter exists then $\mathsf{eval}(\mathsf{Inst}_{\mathsf{bnd}(\mathcal{E})}(\mathcal{E}))$ is a BES.*

Example 2. Consider the following PBES $\mathcal{E}$:

$$(\nu X(b{:}B) = b \wedge Y(\neg b)) \ (\mu Y(b{:}B) = \neg b \vee X(b))$$

Instantiation of X and Y in $\mathcal{E}$ yields the BES $\mathcal{E}'$ below (after minor rewriting):

$$(\nu X_\top = Y_\bot) \ (\nu X_\bot = \bot) \ (\mu Y_\top = X_\top) \ (\mu Y_\bot = \top)$$

A simple Gauß Elimination procedure can solve the BES $\mathcal{E}'$, yielding the following solved BES:

$$(\nu X_\top = \top) \ (\nu X_\bot = \bot) \ (\mu Y_\top = \top) \ (\mu Y_\bot = \top)$$

We have the following correspondence between $\mathcal{E}$ and $\mathcal{E}'$ for arbitrary η and ε:

- $\llbracket \mathcal{E} \rrbracket \eta\varepsilon(X)(b) = \llbracket \mathcal{E}' \rrbracket \eta\varepsilon(X_b) = \llbracket b \rrbracket$ for $b \in B$, and
- $\llbracket \mathcal{E} \rrbracket \eta\varepsilon(Y)(b) = \llbracket \mathcal{E}' \rrbracket \eta\varepsilon(Y_b) = \top$ for $b \in B$. □

4 Instantiation on countable domains

In the previous section, we assumed that the domain D of the instantiated parameter was finite. Instantiation then resulted in a finite PBES in which the predicate variables still carried parameters with a (possibly) infinite domain. In this section, we lift the restriction of finiteness and consider PBESs in which each predicate variable is of type $D \to B$ or of type B, where D is a possibly infinite,

Algorithm 2 The instantiation algorithm for countable domains Inst_∞

$$\mathsf{Inst}_\infty(\epsilon) \quad \triangleq \epsilon$$
$$\mathsf{Inst}_\infty((\sigma X(d{:}D) = \varphi)\ \mathcal{E}) \triangleq \{(\sigma X_v = \mathsf{Sub}_\infty(\varphi[v/d])) \mid v{\in}D\}\ \mathsf{Inst}_\infty(\mathcal{E})$$

where

$$\begin{aligned}
&\mathsf{Sub}_\infty(b) && \triangleq \mathsf{eval}(b)\\
&\mathsf{Sub}_\infty(X(d)) && \triangleq \textstyle\bigvee_{v\in D}\ (\mathsf{eval}(v = d) \wedge X_v)\\
&\mathsf{Sub}_\infty(\varphi_1 \oplus \varphi_2) && \triangleq \mathsf{Sub}_\infty(\varphi_1) \oplus \mathsf{Sub}_\infty(\varphi_2)\\
&\mathsf{Sub}_\infty(\mathsf{Q}d{:}D.\ \varphi) && \triangleq \textstyle\bigoplus_{v\in D} \mathsf{Sub}_\infty(\varphi[v/d]) \qquad \text{where } \textstyle\bigoplus = \bigwedge \text{ if } \mathsf{Q} = \forall, \text{ else } \bigoplus = \bigvee.
\end{aligned}$$

yet countable domain. To each predicate variable $X{:}D \to B$, we associate a countable set of proposition variables $\mathsf{all}(X) \triangleq \{X_d{:}B \mid d{\in}D\}$.
The instantiation algorithm is Algorithm 2; it generates an IBES from a PBES. For every equation $\sigma X(d{:}D) = \varphi$ in the PBES, a block of countably many equations is generated, each of which is of the form $\sigma X_v = \omega_v$ for some $v \in D$ and infinite proposition formula $\omega_v = \mathsf{Sub}_\infty(\varphi[v/d])$. To ensure that every ω_v is indeed a proper infinite proposition formula, we rely on the term evaluator eval to rewrite every data term in $\varphi[v/d]$ to either $\top$ or $\bot$. Hence, $\varphi[v/d]$ must be closed implying that φ may contain no free data variables other than d. This is ensured by allowing only data-closed PBESs for algorithm Inst_∞.

The main correspondence between the predicate variables of a PBES and the proposition variables of the IBES resulting from the instantiation is given in Theorem 2. Below, we first lift Lemma 3 to countable domains.

Lemma 10. *Let φ be a closed predicate formula. Let η_1, η_2 be environments such that $\forall X \in \mathsf{occ}(\varphi) : \forall v \in D : \eta_1(X)([\![v]\!]) = \eta_2(X_v)$. Then for any environment ε:*

$$[\![\varphi]\!]\eta_1\varepsilon = [\![\mathsf{Sub}_\infty(\varphi)]\!]\eta_2.$$

Proof. The proof is similar to the proof of Lemma 3 and therefore omitted. □

Theorem 2. *For all data-closed PBESs $\mathcal{E}$ for which every $X \in \mathsf{var}(\mathcal{E})$ is instantiation-fresh, and all environments η satisfying:*

$$\forall Y \in \mathsf{occ}(\mathcal{E}) \setminus \mathsf{bnd}(\mathcal{E}) : \forall w \in D : \eta(Y_w) = \eta(Y)([\![w]\!]) \tag{1}$$

it holds that, for any environment ε:

$$\forall X \in \mathsf{bnd}(\mathcal{E}) : \forall v \in D : ([\![\mathsf{Inst}_\infty(\mathcal{E})]\!]\eta)(X_v) = ([\![\mathcal{E}]\!]\eta\varepsilon)(X)([\![v]\!]).$$

Proof. Let $\mathcal{E}$ be a data-closed PBES and η, ε be environments such that $\mathcal{E}$ all variables $X \in \mathsf{var}(\mathcal{E})$ are instantiation-fresh, and η satisfies (1). The proof goes by induction on the length of $\mathcal{E}$. If $\mathcal{E} = \epsilon$ the statement holds vacuously. For the inductive case we assume, for all PBESs $\mathcal{E}'$ of length m for which all variables are instantiation-fresh and environments η', ε' satisfying (1):

$$\forall X \in \mathsf{bnd}(\mathcal{E}') : \forall v \in D : ([\![\mathsf{Inst}_\infty(\mathcal{E}')]\!]\eta')(X_v) = ([\![\mathcal{E}']\!]\eta'\varepsilon')(X)([\![v]\!]). \tag{IH}$$

Suppose $\mathcal{E}$ is of length $m+1$, so $\mathcal{E} = (\sigma Y(d{:}D) = \varphi)\ \mathcal{E}'$ for some PBES $\mathcal{E}'$ of length m. We define the following shorthands:

$$\begin{aligned}
\sigma\mathcal{B} &\stackrel{\Delta}{=} \{\sigma Y_w = \mathsf{Sub}_\infty(\varphi[w/d]) \mid w \in D\}\\
f &\stackrel{\Delta}{=} \sigma g \in \mathbb{B}^D.\ \lambda w \in D.\ [\![\mathsf{Sub}_\infty(\varphi[w/d])]\!]([\![\mathsf{Inst}_\infty(\mathcal{E}')]\!]\eta[g/Y_D])\\
h &\stackrel{\Delta}{=} \sigma k \in \mathbb{B}^{\mathbb{D}}.\ \lambda w \in \mathbb{D}.\ [\![\varphi]\!]([\![\mathcal{E}']\!]\eta[k/Y]\varepsilon)\varepsilon[w/d].
\end{aligned}$$

Let $X \in \mathsf{bnd}(\mathcal{E})$ and $v \in D$. Then:

$$\begin{aligned}
&[\![\mathsf{Inst}_\infty((\sigma Y(d{:}D) = \varphi)\ \mathcal{E}')]\!]\eta(X_v)\\
&= [\![\sigma\mathcal{B}\ \mathsf{Inst}_\infty(\mathcal{E}')]\!]\eta(X_v)\\
&= [\![\mathsf{Inst}_\infty(\mathcal{E}')]\!]\eta[f/Y_D](X_v)\\
&= [\![\mathsf{Inst}_\infty(\mathcal{E}')]\!]\eta[f/Y_D][h/Y](X_v)\\
&\stackrel{*}{=} ([\![\mathcal{E}']\!]\eta[f/Y_D][h/Y]\varepsilon)(X)([\![v]\!])\\
&= ([\![\mathcal{E}']\!]\eta[h/Y]\varepsilon)(X)([\![v]\!])\\
&= ([\![(\sigma Y(d{:}D) = \varphi)\ \mathcal{E}']\!]\eta\varepsilon)(X)([\![v]\!]).
\end{aligned}$$

At * we used (IH) for which we need to prove that:

$$\forall X \in \mathsf{var}(\mathcal{E}') : \mathsf{all}(X) \cap \mathsf{var}(\mathcal{E}') = \emptyset \tag{2}$$

$$\forall Z \in \mathsf{occ}(\mathcal{E}') \setminus \mathsf{bnd}(\mathcal{E}') : \forall x \in D : \eta[f/Y_D][h/Y](Z_x) = \eta[f/Y_D][h/Y](Z)([\![x]\!]). \tag{3}$$

Property (2) follows from the facts that all variables in $\mathcal{E}$ are instantiation-fresh and $\mathsf{var}(\mathcal{E}') \subseteq \mathsf{var}(\mathcal{E})$. Regarding (3), let $Z \in \mathsf{occ}(\mathcal{E}') \setminus \mathsf{bnd}(\mathcal{E}')$ and $x \in D$. If $Z \neq Y$ then observe that $Z \in \mathsf{occ}(\mathcal{E}) \setminus \mathsf{bnd}(\mathcal{E})$. Then because $\mathcal{E}$ and η satisfy (1): $\eta[f/Y_D][h/Y](Z_x) = \eta(Z_x) = \eta(Z)([\![x]\!]) = \eta[f/Y_D][h/Y](Z)([\![x]\!])$.

If $Z = Y$ then $\eta[f/Y_D][h/Y](Y_x) = f(x)$ and $\eta[f/Y_D][h/Y](Y)([\![x]\!]) = h([\![x]\!])$, so we need to prove that $f(x) = h([\![x]\!])$. Let $g : D \to \mathbb{B}$ and for that g define $k : \mathbb{D} \to \mathbb{B}$ as follows: $k([\![d]\!]) = g(d)$ for all $d \in D$. Then $f(x) = h([\![x]\!])$ follows if:

$$\begin{aligned}
&(\lambda w \in D.\ [\![\mathsf{Sub}_\infty(\varphi[w/d])]\!]([\![\mathsf{Inst}_\infty(\mathcal{E}')]\!]\eta[g/Y_D]))(x)\\
&= (\lambda w \in \mathbb{D}.\ [\![\varphi]\!]([\![\mathcal{E}']\!]\eta[k/Y]\varepsilon)\varepsilon[w/d])([\![x]\!]).
\end{aligned}$$

We derive, starting at the right-hand side:

$$\begin{aligned}
&(\lambda w \in \mathbb{D}.\ [\![\varphi]\!]([\![\mathcal{E}']\!]\eta[k/Y]\varepsilon)\varepsilon[w/d])([\![x]\!])\\
&= [\![\varphi]\!]([\![\mathcal{E}']\!]\eta[k/Y]\varepsilon)\varepsilon[[\![x]\!]/d]\\
&= [\![\varphi[x/d]]\!]([\![\mathcal{E}']\!]\eta[k/Y]\varepsilon)\varepsilon\\
&= [\![\varphi[x/d]]\!]([\![\mathcal{E}']\!]\eta[g/Y_D][k/Y]\varepsilon)\varepsilon\\
&\stackrel{**}{=} [\![\mathsf{Sub}_\infty(\varphi[x/d])]\!]([\![\mathsf{Inst}_\infty(\mathcal{E}')]\!]\eta[g/Y_D][k/Y])\\
&= [\![\mathsf{Sub}_\infty(\varphi[x/d])]\!]([\![\mathsf{Inst}_\infty(\mathcal{E}')]\!]\eta[g/Y_D])\\
&= (\lambda w \in D.\ [\![\mathsf{Sub}_\infty(\varphi[w/d])]\!]([\![\mathsf{Inst}_\infty(\mathcal{E}')]\!]\eta[g/Y_D]))(x).
\end{aligned}$$

For convenience we define $\theta \stackrel{\Delta}{=} \eta[g/Y_D][k/Y]$. At ** we used Lemma 10 which is allowed because $\varphi[x/d]$ is closed by data-closedness of $\mathcal{E}$ and we have:

$$\forall W \in \mathsf{occ}(\varphi) : \forall w \in D : ([\![\mathcal{E}']\!]\theta\varepsilon)(W)([\![w]\!]) = ([\![\mathsf{Inst}_\infty(\mathcal{E}')]\!]\theta)(W_w)$$

which we now prove. Let $W \in \mathsf{occ}(\varphi)$ and $w \in D$. There are three cases:

1. $W = Y$. Then $([\![\mathcal{E}']\!]\theta\varepsilon)(Y)([\![w]\!]) = k([\![w]\!]) = g(w) = ([\![\mathsf{Inst}_\infty(\mathcal{E}')]\!]\theta)(Y_w)$.
2. $W \neq Y$ and $W \notin \mathsf{bnd}(\mathcal{E})$. Then:

$$([\![\mathcal{E}']\!]\theta\varepsilon)(W)([\![w]\!]) = \eta(W)([\![w]\!]) \stackrel{\dagger}{=} \eta(W_w) \stackrel{\ddagger}{=} ([\![\mathsf{Inst}_\infty(\mathcal{E}')]\!]\theta)(W_w).$$

   At $\dagger$ we used the fact that $\mathcal{E}$ and η satisfy (1) in combination with $W \in \mathsf{occ}(\mathcal{E}) \setminus \mathsf{bnd}(\mathcal{E})$. At $\ddagger$ we used Lemma 1 combined with $W_w \notin \mathsf{bnd}(\mathsf{Inst}_\infty(\mathcal{E}'))$.
3. $W \neq Y$ and $W \in \mathsf{bnd}(\mathcal{E})$. Observe that $W \in \mathsf{bnd}(\mathcal{E}')$. Then the equivalence follows from (IH) if we prove that all variables in $\mathcal{E}'$ are instantiation-fresh and θ satisfies (1). Because all variables in $\mathcal{E}$ are instantiation-fresh and $\mathsf{var}(\mathcal{E}') \subseteq \mathsf{var}(\mathcal{E})$, we have that all variables in $\mathcal{E}'$ are instantiation-fresh. The fact that $\mathcal{E}'$ and θ satisfy (1) follows using similar reasonings as in cases 1 and 2. □

As a corollary of the above Theorem, we have the following result:

Corollary 2. *Let $\mathcal{E}$ be a PBES. If $\mathcal{E}$ is closed and data-closed, and all data sorts occurring in $\mathcal{E}$ are countable and a suitable term rewriter* eval *exists then $\mathsf{Inst}_\infty(\mathcal{E})$ is an IBES.*

Remark 1. For typical verification problems, such as (local) model checking and equivalence checking, a partial solution to the PBES is often satisfactory. Using Proposition 1, it is straightforward to turn the instantiation scheme for PBESs involving countable data sorts into a procedure for computing a BES. This can be achieved by, *e.g.*, an on-the-fly depth-first or breadth-first exploration of all the equations for the required (instantiated) binding variables of the theoretical IBES. A similar technique is discussed in [10], and, thence, we do not further explore this issue here.

Example 3. Consider the PBES $\mathcal{E}$, consisting of a single equation, given below:

$$(\nu X(n{:}N) = n \neq 1 \wedge X(n+1))$$

Applying the transformation Inst_∞, we obtain the following IBES:

$$\{(\nu X_0 = X_1)\ (\nu X_1 = \bot)\ (\nu X_n = X_{n+1}) \mid n \geq 2\}$$

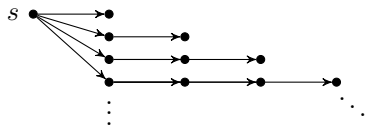
While solving *e.g.* $X(5)$ by means of a transformation to IBES would require an infinite computation, solving $X(0)$ or $X(1)$ would terminate using a local resolution, since *e.g.* $X(0)$ depends on $X(1)$ which is immediately false. □

Remark 2. The transformation from an IBES to a PBES is elementary, provided one has a sufficiently rich data language: the sorts that are used for the blocks in the IBES can be introduced as the data sorts of the PBES, and the infinite disjunction and conjunction that occur in the infinite proposition formulae can be converted to equality tests and universal and existential quantifications, respectively.

5 Examples

In this section, we provide further demonstrations of the various uses of the basic instantiation techniques of the previous sections. First, we demonstrate that the partial instantiation of a PBES is a powerful manipulation in itself. Two prime –but lengthy– example applications of the manipulation are already contained in the full version of [3]. Two smaller examples, derived from model checking problems on infinite state systems, are given below. These problems first appeared in [2] and were repeated in [8] to demonstrate the power of IBESs.

Example 4. Consider the infinite transition system depicted below. The property that Bradfield [2] and Mader [8] verify is that every path that starts in s has only finite length, a property that is given by the following modal μ-calculus formula: $\mu X.[-]X$. Notice that the number of paths in the system is infinite.



The following PBES, consisting of a single fixpoint equation, encodes the above model checking problem, where satisfaction of the property by state s corresponds to $X(\top, 0)$:[1]

$$\mu X(b{:}B, n{:}N) = (\forall i{:}N.\ \neg b \vee X(\neg b, i)) \wedge (b \vee n = 0 \vee X(b, n-1)).$$

A solution technique based on a straightforward symbolic approximation as described in *e.g.* [7] does not terminate. Patterns [6] for solving PBESs, which allow one to "look up" a solution for equations of a particular shape, are also not applicable. Instantiation of Booleans leads to the following PBES:

$$(\mu X_\bot(n{:}N) = (n = 0) \vee X_\bot(n-1)) \quad (\mu X_\top(n{:}N) = \forall i{:}N.\ X_\bot(i)).$$

The equation for $X_\bot$ can easily be solved by means of a pattern, immediately leading to the following equivalent equation system:

$$(\mu X_\bot(n{:}N) = \exists i{:}N.\ n = i) \quad (\mu X_\top(n{:}N) = \forall i{:}N.\ X_\bot(i)).$$

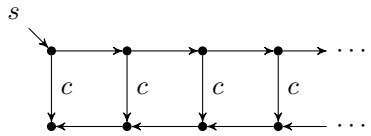
Using standard logic, the above equation system can immediately be rewritten (even automatically [6]) to the following:

$$(\mu X_\bot(n{:}N) = \top) \quad (\mu X_\top(n{:}N) = \top).$$

Hence the property holds. The proof in [8] requires a manual construction of a set-based representation of an IBES, and requires showing the well-foundedness of mappings of this representation. The tableau-based methods of [2] require the investigation of *extended paths*. The proof strategy we followed is easier to understand and requires less effort. □

[1] Notice that the PBES can be obtained fully automatically, for details, see [7]. For an implementation, see a.o. the tool *lps2pbes* of mCRL2 [http://mcrl2.org].

Example 5. The below infinite transition system originated from a Petri Net given by Bradfield [2], and reappeared in Mader [8] as a transition system. We copy the latter representation. The property that is verified is that the every behaviour in the transition system exhibits only a finite number of c actions: $\mu X.\nu Y.([c]X \wedge [\neg c]Y)$; note that the formula is of alternation depth 2.



The following PBES, consisting of two fixpoint equations, encodes the above model checking problem, where $s \models \mu X.\nu Y.\ ([c]X \wedge [\neg c]Y)$ is encoded by $X(\top, 0)$:

$(\mu X(b{:}B, n{:}N) = Y(b, n))$
$(\nu Y(b{:}B, n{:}N) = (\neg b \vee X(\neg b, n)) \wedge (\neg b \vee Y(b, n+1)) \wedge (b \vee n{=}0 \vee Y(b, n-1)))$

The above equation system can only be solved by a complex pattern, described in [7], which requires one to introduce an auxiliary selector function. Instantiation of Booleans for equation Y leads to the following PBES:

$$(\mu X(b{:}B, n{:}N) = (b \wedge Y_\top(n)) \vee (\neg b \wedge Y_\bot(n)))$$
$$(\nu Y_\top(n{:}N) = X(\bot, n) \wedge Y_\top(n+1))$$
$$(\nu Y_\bot(n{:}N) = n{=}0 \vee Y_\bot(n-1)).$$

The equation for $Y_\bot$ can easily be solved by means of a pattern [7]; the equation for $Y_\top$ is solved instantly using symbolic approximation. This leads to the following equivalent equation system:

$$(\mu X(b{:}B, n{:}N) = (b \wedge Y_\top(n)) \vee (\neg b \wedge Y_\bot(n)))$$
$$(\nu Y_\top(n{:}N) = X(\bot, n)) \quad (\nu Y_\bot(n{:}N) = \exists i{:}N.\ n - i{=}0).$$

Using standard logic, the equation for $Y_\bot$ can be simplified to $\top$; the solution to $Y_\bot$ and $Y_\top$ can be substituted in the equation for X, leading to:

$$(\mu X(b{:}B, n{:}N) = (b \wedge X(\bot, n)) \vee \neg b).$$

Simplification using symbolic approximation yields the solution $\top$ as the third approximant. Hence the property holds vacuously. Again, the proof using partial instantiation is straightforward and enables the use of simple pattern matching, while the proofs by Mader and Bradfield require more effort. □

Next, we illustrate the feasibility of instantiating a given PBES to (I)BES. The below examples are model checking problems on various systems; most of these systems rely on natural numbers to model some of their aspects. A full instantiation of the PBES would therefore yield an IBES, but the local resolution yields a BES in all cases.

Example 6. A prototype tool[2] has been implemented that instantiates a given PBES, and, upon termination has computed a BES that holds the answer to whether a particular equation in the original PBES is true for some data value. The table below shows the time performance of this tool[3] on several publicly available benchmarks, consisting of industrial protocols and systems (first four) and games (second three). The property encoded in the PBES was absence of deadlock, which would require all reachable states to be computed. This allows for a fair comparison with explicit state space generating tools. Of course, more involved properties can also be encoded, *e.g.* fairness and liveness properties. Such more involved properties often yield PBESs of higher alternation depth. The deadlock property yields an alternation-free PBES.

	BRP	IEEE-1394	car-lift	chatbox	domineering	clobber	othello
States (#)	10,548	188,569	4,312	65,536	455,317	600,161	55,093
Transitions (#)	12,168	340,607	9,918	2,162,688	2,062,696	2,221,553	88,258
LTS (sec)	7	215	14	37	78	148	159
BES (sec)	6	232	10	15	82	182	186

We added the performance of the tool that would generate the state space. Our performance is in general comparable; differences are attributed to minor differences in rewriting strategies. Our experience with memory usage is similar. Note that the BES generation time includes solving the BES using an off-the-shelf BES-solving technique. □

Example 7. The previous example illustrated the efficacy of full instantiation of PBESs to (I)BESs for alternation-free PBESs. We next consider PBESs that encode the branching bisimulation equivalence problem [3]. This encoding yields a PBES with alternation depth 2 (an alternative encoding exists that yields an alternation-free PBES for finite systems). The table below shows the time performance of our prototype tool on instantiating three PBESs encoding the equivalence between the *Concurrent Alternating Bit Protocol* (CABP), the *Alternating Bit Protocol* (ABP) and the *One-Place Buffer* (OPB). Each protocol has ten different messages.

	OPB	ABP	CABP	ABP ≈ OPB	CABP ≈ OPB	ABP ≈ CABP
States (#)	11	362	3,536			
Transitions (#)	20	460	13,791			
BES (# equations)				2,884	35,104	268,064
BES (sec)				< 1	2	13

□

6 Conclusions

Parameterised Boolean Equation Systems have demonstrated to be quite suited for studying various formal verification problems. Several unique solution techniques for solving PBESs have been studied and shown to be effective.

[2] The tool is part of the mCRL2 toolset (revision 4413) and is called *pbes2bool*.
[3] All experiments were run on a Macbook, 2 Gb, Intel C2D (2Ghz) running Linux.

To this set of solution techniques, we have added a new set of manipulations, which admit a wider class of PBESs to be solved either automatically or by means of (syntactic) manipulations. From the point of view of the basic theory of PBESs, the manipulations firmly relate PBESs to two other prominent notions of equation systems, viz. BESs and IBESs. This provides a different angle on the somewhat complex basic theory of PBESs.

We have also reported on the efficacy of our manipulations (see Section 5). Among others, we report on the results of typical verification problems conducted using a prototype tool. The tool implements the transformation from PBESs to (I)BESs that is required for a local resolution of the PBES. In view of this, our prototype tooling, and the examples in Section 5 demonstrate the feasibility of the approach outlined in [10].

Acknowledgements. We thank Wieger Wesselink and Jan Friso Groote for valuable feedback on the prototype implementation.

References

1. H. Bekič. *Programming Languages and Their Definition.* LNCS 177. Springer-Verlag, 1984.
2. J.C. Bradfield. *Verifying Temporal Proporties of Systems.* Birkhäuser, 1992.
3. 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. Full version appeared as CSR 07-14, Eindhoven University of Technology.
4. 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.
5. 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.
6. J.F. Groote and T.A.C. Willemse. Model-checking processes with data. *Sci. Comput. Program*, 56(3):251–273, 2005.
7. J.F. Groote and T.A.C. Willemse. Parameterised boolean equation systems. *Theor. Comput. Sci*, 343(3):332–369, 2005.
8. A. Mader. *Verification of Modal Properties Using Boolean Equation Systems.* PhD thesis, Technische Universität München, 1997.
9. A. Mader. Verification of modal properties using infinite boolean equation systems. Technical Report CSI-R9727, University of Nijmegen, Nijmegen, 1997.
10. R. Mateescu. Local model-checking of an alternation-free value-based modal mu-calculus. In *Proc. 2nd Int'l Workshop on VMCAI*, September 1998.
11. A. Tarski. A lattice-theoretical fixpoint theorem and its applications. *Pacific J. Mathematics*, 5(2):285–309, June 1955.