Equivalence Checking for Infinite Systems Using Parameterized Boolean Equation Systems

Taolue Chen[1,⋆], Bas Ploeger[2,⋆⋆], Jaco van de Pol[1,2], and Tim A.C. Willemse[2,⋆⋆⋆]

[1] CWI, Department of Software Engineering,
P.O. Box 94079, 1090 GB Amsterdam, The Netherlands
[2] Eindhoven University of Technology, Design and Analysis of Systems Group,
P.O. Box 513, 5600 MB Eindhoven, The Netherlands

Abstract. In this paper, we provide a transformation from the branching bisimulation problem for infinite, concurrent, data-intensive systems in linear process format, into solving Parameterized Boolean Equation Systems. We prove correctness, and illustrate the approach with an unbounded queue example. We also provide some adaptations to obtain similar transformations for weak bisimulation and simulation equivalence.

1 Introduction

A standard approach for verifying the correctness of a computer system or a communication protocol is the *equivalence-based methodology*. This framework was introduced by Milner [23] and has been intensively explored in process algebra. One proceeds by establishing two descriptions (models) for one system: a *specification* and an *implementation*. The former describes the desired high-level behavior, while the latter provides lower-level details indicating how this behavior is to be achieved. Then an implementation is said to be correct, if it behaves "the same as" its specification. Similarly, one could check whether the implementation has "at most" the behavior allowed by the specification. Several *behavioral equivalences and preorders* have been introduced to relate specifications and implementations, supporting different notions of observability. These include strong, weak [24], and branching bisimulation [11,4].

Equivalence Checking for Finite Systems. Checking strong bisimulation of finite systems can be done very efficiently. The basic algorithm is the well-known *partition refinement* algorithm [26]. For weak bisimulation checking, one could compute the transitive closure of τ-transitions, and thus lift the algorithms for strong bisimulation to the weak one. This is viable but costly, since it might incur a quadratic blow-up w.r.t.

⋆ This author is partially supported by Dutch Bsik project BRICKS, 973 Program of China (2002CB312002), NSF of China (60233010, 60273034, 60403014), 863 Program of China (2005AA113160, 2004AA112090).

⋆⋆ This author is partially supported by the Netherlands Organisation for Scientific Research (NWO) under VoLTS grant number 612.065.410.

⋆⋆⋆ This author is partially supported by the Netherlands Organisation for Scientific Research (NWO) under BRICKS/FOCUS grant number 642.000.602.

L. Caires and V.T. Vasconcelos (Eds.): CONCUR 2007, LNCS 4703, pp. 120–135, 2007.

the original LTSs. Instead, one could employ the more efficient solution by [15] for checking branching bisimulation, as branching and weak bisimulation often coincide.

Alternatively, one can transform several bisimulation relations into Boolean Equation Systems (BES). Various encodings have been proposed in the literature [2,8,22], leading to efficient tools. In [2] it is shown that the BESs obtained from equivalence relations have a special format; the encodings of [22] even yield alternation free BESs (cf. definition of alternation depth in [21]) for up to five different behavioral equivalences. Solving alternation free BESs can be done very efficiently. However, finiteness of the graphs is crucial for the encodings yielding alternation free BESs.

It is interesting to note that the μ-calculus model checking problem for finite systems can also be transformed to the problem of solving a BES [2,21]. Hence, a BES solver, e.g. [22], provides a uniform engine for verification by model checking and equivalence checking for finite systems.

Our Contribution. In this paper, we focus on equivalence checking for *infinite* systems. Generally for concurrent systems with data, the induced labeled transition system (LTS) is no longer *finite*, and the traditional algorithms fail for *infinite* transition graphs. The symbolic approach needed for infinite systems depends on the specification format. We use *Linear Process Equations* (LPEs), which originate from μCRL [14], a process algebra with abstract data types, and describe the system by a finite set of guarded, nondeterministic transitions. LPEs are Turing complete, and many formalisms can be compiled to LPEs without considerable blow-up. Therefore, our methods essentially also apply to LOTOS [5], timed automata [1], I/O-automata [20], finite control π-calculus [25], UNITY [6], etc.

The solution we propose in this paper is inspired by [12], where the question whether an LPE satisfies a *first-order* μ-calculus formula is transformed into a *Parameterized Boolean Equation System* (PBES). PBESs extend boolean equation systems with data parameters and quantifiers. Heuristics, techniques [17], and tool support [16] have been developed for solving PBESs. This is still subject to ongoing research. Also in [28] such equation systems are used for model checking systems with data and time. In general, solving PBESs cannot be completely automated.

We propose to check branching bisimilarity of infinite systems by solving recursive equations. In particular, we show how to generate a PBES from two LPEs. The resulting PBES has alternation depth two. We prove that the PBES has a positive solution if and only if the two (infinite) systems are branching bisimilar. Moreover, we illustrate the technique by an example on unbounded queues, and show similar transformations for Milner's weak bisimulation [24] and branching simulation equivalence [10].

There are good reasons to translate branching bisimulation for infinite systems to solving PBESs, even though both problems are undecidable. The main reason is that solving PBESs is a more fundamental problem, as it boils down to solving equations between predicates. The other reason is that model checking mu-calculus with data has already been mapped to PBESs. Hence all efforts in solving PBESs (like [17]) can now be freely applied to the bisimulation problem as well.

Related Work. We already mentioned related work on finite systems, especially [2,22]. There are several approaches on which we want to comment in more detail.

The cones and foci method [9] rephrases the question whether two LPEs are bisimilar in terms of proof obligations on data objects. Basically, the user must first identify invariants, a focus condition, and a state mapping. In contrast, generating a PBES requires no human ingenuity, although solving the PBES still may. Furthermore, our solution is considerably more general, because it lifts two severe limitations of the cones and foci method. The first limitation is that the cones and foci method only works in case the branching bisimulation is functional (this means that a state in the implementation can only be related to a unique state in the specification). Another severe limitation of the cones and foci method is that it cannot handle specifications with τ-transitions. In some protocols (e.g. the bounded retransmission protocol [13]) this condition is not met and thus the cones and foci method fails. In our example on unbounded queues, both systems perform τ steps, and their bisimulation is not functional.

Our work can be seen as the generalization of [19] to weak and branching equivalences. In [19], Lin proposes Symbolic Transition Graphs with Assignments (STGA) as a new model for message-passing processes. An algorithm is also presented which computes bisimulation formulae for finite state STGAs, in terms of the greatest solutions of a *predicate equation system*. This corresponds to an alternation free PBES, and thus it can only deal with strong bisimulation.

The extension of Lin's work for strong bisimulation to weak and branching equivalences is not straightforward. This is testified by the encoding of weak bisimulation in predicate systems by Kwak *et al.* [18]. However, their encoding is not generally correct for STGA, as they use a conjunction over the complete τ-closure of a state. This only works in case that the τ-closure of every state is finite, which is generally not the case for STGA, also not for our LPEs. Alternation depth 2 seems unavoidable but does not occur in [18]. Note that for finite LTS a conjunction over the τ-closure is possible [22], but leads to a quadratic blow-up of the BES in the worst case.

Structure of the Paper. The paper is organized as follows. In Section 2, we provide background knowledge on linear process equations, labeled transition systems and bisimulation equivalences. We assume familiarity with standard fixpoint theory. In Section 3, PBESs are reviewed. Section 4 is devoted to the presentation of the translation and the justification of its correctness. In Section 5, we provide an example to illustrate the use of our algorithm. In Section 6, we demonstrate how to adapt the translation for branching bisimulation to weak bisimulations and simulation equivalence. The translation for strong bisimulation and an additional example are presented in [7]. The paper is concluded in Section 7.

2 Preliminaries

Linear process equations have been proposed as a *symbolic* representation of general (infinite) labeled transition systems. In an LPE, the behavior of a process is denoted as a state vector of typed variables, accompanied by a set of condition-action-effect rules. LPEs are widely used in μCRL [14], a language for specifying concurrent systems and protocols in an algebraic style. We mention that μCRL has complete automatic tool support to generate LPEs from μCRL specifications.

Definition 1 (Linear Process Equation). *A* linear process equation *is a parameterized equation taking the form*

$$M(d:D) = \sum_{a \in Act} \sum_{e_a : E_a} h_a(d, e_a) \implies a(f_a(d, e_a)) \cdot M(g_a(d, e_a))$$

where $f_a : D \times E_a \to D_a$, $g_a : D \times E_a \to D$ and $h_a : D \times E_a \to \mathbb{B}$ for each $a \in Act$. Note that here D, D_a and E_a are general data types and $\mathbb{B}$ is the boolean type.

In the above definition, the LPE M specifies that if in the current state d the condition $h_a(d, e_a)$ holds for any e_a of sort E_a, then an action a carrying data parameter $f_a(d, e_a)$ is possible and the effect of executing this action is the new state $g_a(d, e_a)$. The values of the condition, action parameter and new state may depend on the current state and a summation variable e_a.

For simplicity and without loss of generality, we restrict ourselves to a single variable at the left-hand side in all our theoretical considerations and to the use of non-terminating processes. That is, we do not consider processes that, apart from executing an infinite number of actions, also have the possibility to perform a finite number of actions and then terminate successfully. Including multiple variables and termination in our theory does not pose any theoretical challenges, but is omitted from our exposition for brevity. The operational semantics of LPEs is defined in terms of *labeled transition systems*.

Definition 2 (Labeled Transition System). *The* labeled transition system *of an LPE (as defined in Definition 1) is a quadruple $\mathcal{M} = \langle \mathcal{S}, \Sigma, \to, s_0 \rangle$, where*

- *$\mathcal{S} = \{d \mid d \in D\}$ is the (possibly infinite) set of states;*
- *$\Sigma = \{a(d) \mid a \in Act \wedge d \in D_a\}$ is the (possibly infinite) set of labels;*
- *$\to = \{(d, a(d'), d'') \mid a \in Act \wedge \exists e_a \in E_a . h_a(d, e_a) \wedge d' = f_a(d, e_a) \wedge d'' = g_a(d, e_a)\}$ is the transition relation;*
- *$s_0 = d_0 \in \mathcal{S}$, for a given $d_0 \in D$, is the initial state.*

For an LPE M, we usually write $d \xrightarrow{a(d')}_M d''$ to denote the fact that $(d, a(d'), d'')$ is in the transition relation of the LTS of M. We will omit the subscript M when it is clear from the context. Following Milner [24], the derived transition relation $\Rightarrow$ is defined as the reflexive, transitive closure of $\xrightarrow{\tau}$ (i.e. $(\xrightarrow{\tau})^*$), and $\overset{\alpha}{\Rightarrow}$, $\overset{\hat{\alpha}}{\Rightarrow}$ and $\xrightarrow{\bar{\alpha}}$ are defined in the standard way as follows:

$$\overset{\alpha}{\Rightarrow} \overset{\text{def}}{=} \Rightarrow\!\xrightarrow{\alpha}\!\Rightarrow \qquad \overset{\hat{\alpha}}{\Rightarrow} \overset{\text{def}}{=} \begin{cases} \Rightarrow \text{ if } \alpha = \tau \\ \overset{\alpha}{\Rightarrow} \text{ otherwise.} \end{cases} \qquad \xrightarrow{\bar{\alpha}} \overset{\text{def}}{=} \begin{cases} \xrightarrow{\tau} \cup \mathrm{Id} \text{ if } \alpha = \tau \\ \xrightarrow{\alpha} \quad \text{otherwise.} \end{cases}$$

2.1 Bisimulation Equivalences

We now introduce several well-known equivalences. The definitions below are with respect to an arbitrary, given labeled transition system $\mathcal{M} = \langle S, \Sigma, \to, s_0 \rangle$.

Definition 3 (Branching (Bi)simulations). *A binary relation $\mathcal{R} \subseteq S \times S$ is a* semi-branching simulation*, iff whenever $s \mathcal{R} t$ then for all $\alpha \in \Sigma$ and $s' \in S$, if $s \xrightarrow{\alpha} s'$, then $t \Rightarrow t' \xrightarrow{\bar{\alpha}} t''$ for some $t', t'' \in S$ such that $s \mathcal{R} t'$ and $s' \mathcal{R} t''$. We say that:*

- $\mathcal{R}$ *is a* semi-branching bisimulation*, if both* $\mathcal{R}$ *and* $\mathcal{R}^{-1}$ *are semi-branching simulations.*
- s *is* branching bisimilar *to* t*, denoted by* $s \underline{\leftrightarrow}_b t$*, iff there exists a semi-branching bisimulation* $\mathcal{R}$*, such that* $s\mathcal{R}t$*.*
- s *is* branching simulation equivalent *to* t*, iff there exist* $\mathcal{R}$ *and* $\mathcal{Q}$*, such that* $s\mathcal{R}t$ *and* $t\mathcal{Q}s$ *and both* $\mathcal{R}$ *and* $\mathcal{Q}$ *are semi-branching simulations.*

Note that although a semi-branching simulation is not necessarily a branching simulation, it is shown in [4] that this definition of branching bisimilarity coincides with the original definition in [11]. Therefore, in the sequel we take the liberty to use *semi-branching* and *branching* interchangeably. In the theoretical considerations in this paper, semi-branching relations are more convenient as they allow for shorter and clearer proofs of our theorems.

Definition 4 (Weak Bisimulation). *A binary relation* $\mathcal{R} \subseteq S \times S$ *is an (early)* weak bisimulation*, iff it is symmetric and whenever* $s\mathcal{R}t$ *then for all* $\alpha \in \Sigma$ *and* $s' \in S$*, if* $s \xrightarrow{\alpha} s'$*, then* $t \stackrel{\hat{\alpha}}{\Rightarrow} t'$ *for some* $t' \in S$ *such that* $s'\mathcal{R}t'$*.*
Weak bisimilarity*, denoted by* $\underline{\leftrightarrow}_w$*, is the largest weak bisimulation.*

3 Parameterized Boolean Equation Systems

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

$$\sigma X(d : D) = \phi$$

σ denotes either the minimal (μ) or the maximal (ν) fixpoint. X is a predicate variable (from a set $\mathcal{P}$ of predicate variables) that binds a data variable d (from a set $\mathcal{D}$ of data variables) that may occur freely in the *predicate formula* ϕ. Apart from data variable d, ϕ can contain data terms, boolean connectives, quantifiers over (possibly infinite) data domains, and predicate variables. Predicate formulae ϕ are formally defined as follows:

Definition 5 (Predicate Formula). *A* predicate formula *is a formula* ϕ *in positive form, defined by the following grammar:*

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

where b *is a data term of sort* $\mathbb{B}$*, possibly containing data variables* $d \in \mathcal{D}$*. Furthermore,* $X \in \mathcal{P}$ *is a (parameterized) predicate variable and* e *is a data term.*

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 $\mathbb{B}$.

The semantics of predicates is dependent on the semantics of data terms. For a closed term e, we assume an interpretation function $[\![e]\!]$ that maps e to the data element it represents. For open terms, we use a *data environment* ε that maps each variable from $\mathcal{D}$ to a data value of the right sort. The interpretation of an open term e is denoted as $[\![e]\!]\varepsilon$ in the standard way.

Definition 6 (Semantics). *Let $\theta : \mathcal{P} \to \wp(D)$ be a* predicate environment *and $\varepsilon : \mathcal{D} \to D$ be a* data environment*. The interpretation of a predicate formula ϕ in the context of environment θ and ε, written as $[\![\phi]\!]\theta\varepsilon$, is either true or false, determined by the following induction:*

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

Definition 7 (Parameterized Boolean Equation System). *A* parameterized boolean equation system *is a finite sequence of equations of the form $\sigma X(d : D) = \phi$ where ϕ is a predicate formula in which at most d may occur as a free data variable. The empty equation system is denoted by ϵ.*

In the remainder of this paper, we abbreviate parameterized 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. The *solution* to an equation system is defined in the context of a predicate environment, as follows.

Definition 8 (Solution to an Equation System). *Given a predicate environment θ and an equation system $\mathcal{E}$, the* solution *$[\![\mathcal{E}]\!]\theta$ to $\mathcal{E}$ is an environment that is defined as follows, where σ is the greatest or least fixpoint, defined over the complete lattice $\wp(D)$.*

$$\begin{array}{ll}
[\![\epsilon]\!]\theta & = \theta \\
[\![(\sigma X(d : D) = \phi)\mathcal{E}]\!]\theta & = [\![\mathcal{E}]\!](\theta\Big[\sigma\mathcal{X} \in \wp(D).\lambda v \in D.[\![\phi]\!]([\![\mathcal{E}]\!]\theta[\mathcal{X}/X])[v/d]/X\Big])
\end{array}$$

For *closed* equation systems, the solution for the binding predicate variables does not depend on the given environment θ. In such cases, we refrain from writing the environment explicitly.

4 Translation for Branching Bisimulation

We define a translation that encodes the problem of finding the largest branching bisimulation in the problem of solving an equation system.

Definition 9. *Let M and S be LPEs of the following form:*

$$M(d : D^{\mathsf{M}}) = \sum_{a \in Act} \sum_{e_a : E_a^{\mathsf{M}}} h_a^{\mathsf{M}}(d, e_a) \implies a(f_a^{\mathsf{M}}(d, e_a)).M(g_a^{\mathsf{M}}(d, e_a))$$

$$S(d : D^{\mathsf{S}}) = \sum_{a \in Act} \sum_{e_a : E_a^{\mathsf{S}}} h_a^{\mathsf{S}}(d, e_a) \implies a(f_a^{\mathsf{S}}(d, e_a)).S(g_a^{\mathsf{S}}(d, e_a))$$

Given initial states $d : D^{\mathsf{M}}$ and $d' : D^{\mathsf{S}}$, the equation system that corresponds to the branching bisimulation between LPEs $M(d)$ and $S(d')$ is constructed by the function $brbisim$ (see Algorithm 1).

The main function $brbisim$ returns an equation system in the form $\nu E_2 \mu E_1$ where the bound predicate variables in E_2 are denoted by X and that in E_1 are denoted by Y. Intuitively, E_2 is used to characterize the (branching) bisimulation while E_1 is used to absorb the τ actions. The equation system's predicate formulae are constructed from the syntactic ingredients from LPEs M and S. Note that although we talk about the model (M) and the specification (S), the two systems are treated completely symmetrically. As we will show in Theorem 2, the solution for $X^{\mathsf{M},\mathsf{S}}$ in the resulting equation system gives the largest branching bisimulation relation between M and S as a predicate on $D^{\mathsf{M}} \times D^{\mathsf{S}}$.

Algorithm 1. Generation of a PBES for Branching Bisimulation

brbisim= $\nu E_2 \mu E_1$, **where**

$$E_2 := \{X^{\mathsf{M},\mathsf{S}}(d : D^{\mathsf{M}}, d' : D^{\mathsf{S}}) = match^{\mathsf{M},\mathsf{S}}(d, d') \wedge match^{\mathsf{S},\mathsf{M}}(d', d)\ ,\\ X^{\mathsf{S},\mathsf{M}}(d' : D^{\mathsf{S}}, d : D^{\mathsf{M}}) = X^{\mathsf{M},\mathsf{S}}(d, d')\ \}$$

$$E_1 := \{Y_a^{\mathsf{p},\mathsf{q}}(d : D^{\mathsf{p}}, d' : D^{\mathsf{q}}, e : E_a^{\mathsf{p}}) = close_a^{\mathsf{p},\mathsf{q}}(d, d', e) \mid a \in Act \wedge (\mathsf{p},\mathsf{q}) \in \{(\mathsf{M},\mathsf{S}),(\mathsf{S},\mathsf{M})\}\}$$

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

$$match^{\mathsf{p},\mathsf{q}}(d : D^{\mathsf{p}}, d' : D^{\mathsf{q}}) = \bigwedge_{a \in Act} \forall e : E_a^{\mathsf{p}}.\ (h_a^{\mathsf{p}}(d, e) \implies Y_a^{\mathsf{p},\mathsf{q}}(d, d', e));$$

$$close_a^{\mathsf{p},\mathsf{q}}(d : D^{\mathsf{p}}, d' : D^{\mathsf{q}}, e : E_a^{\mathsf{p}}) = \exists e' : E_\tau^{\mathsf{q}}.\ (h_\tau^{\mathsf{q}}(d', e') \wedge Y_a^{\mathsf{p},\mathsf{q}}(d, g_\tau^{\mathsf{q}}(d', e'), e)) \vee (X^{\mathsf{p},\mathsf{q}}(d, d') \wedge step_a^{\mathsf{p},\mathsf{q}}(d, d', e));$$

$$step_a^{\mathsf{p},\mathsf{q}}(d : D^{\mathsf{p}}, d' : D^{\mathsf{q}}, e : E_a^{\mathsf{p}}) = (a = \tau \wedge X^{\mathsf{p},\mathsf{q}}(g_\tau^{\mathsf{p}}(d, e), d')) \vee \exists e' : E_a^{\mathsf{q}}.\ h_a^{\mathsf{q}}(d', e') \wedge (f_a^{\mathsf{p}}(d, e) = f_a^{\mathsf{q}}(d', e')) \wedge X^{\mathsf{p},\mathsf{q}}(g_a^{\mathsf{p}}(d, e), g_a^{\mathsf{q}}(d', e'));$$

4.1 Correctness of Transformation

In this section we confirm the relation between the branching bisimulation problem and the problem of solving an equation system. Before establishing the correctness of the transformation presented above, we first provide a fixpoint characterization for (semi-) branching bisimilarity, which we exploit in the correctness proof of our algorithm. For brevity, given any LPEs M and S, and any binary relation $\mathcal{B}$ over $D^{\mathsf{M}} \times D^{\mathsf{S}}$, we define a functional $\mathcal{F}$ as

$$\mathcal{F}(\mathcal{B}) = \{(d, d') \mid \forall a \in Act, e_a \in E_a^{\mathsf{M}}.h_a^{\mathsf{M}}(d, e_a) \implies \exists d'_2, d'_3.d' \Rightarrow_S d'_2 \wedge d'_2 \xrightarrow{\overline{a(f_a^{\mathsf{M}}(d,e_a))}}_S d'_3 \wedge (d, d'_2) \in \mathcal{B} \wedge (g_a^{\mathsf{M}}(d, e_a), d'_3) \in \mathcal{B},\\ \text{and } \forall a \in Act, e'_a \in E_a^{\mathsf{S}}.h_a^{\mathsf{S}}(d', e'_a) \implies \exists d_2, d_3.d \Rightarrow_M d_2 \wedge d_2 \xrightarrow{\overline{a(f_a^{\mathsf{S}}(d',e'_a))}}_M d_3 \wedge (d_2, d') \in \mathcal{B} \wedge (d_3, g_a^{\mathsf{S}}(d', e'_a)) \in \mathcal{B}\}$$

It is not difficult to see that $\mathcal{F}$ is monotonic. We claim that branching bisimilarity is the *maximal* fixpoint of functional $\mathcal{F}$ (i.e. $\nu B.\mathcal{F}(B)$).

Lemma 1. $\underline{\leftrightarrow}_b = \nu B.\mathcal{F}(B)$.

Proof. We prove set inclusion both ways using the definition of $\mathcal{F}$ and fixpoint theorems. The full proof is included in [7]. □

For proving the correctness of our translation, we first solve μE_1 given an arbitrary solution for X.

Theorem 1. *For any LPEs M and S, let μE_1 be generated by Algorithm 1, let η be an arbitrary predicate environment, and let $\theta = [\![\mu E_1]\!]\eta$. Then for any action a, and any d, d' and e, we have $(d, d', e) \in \theta(Y_a^{\mathsf{M},\mathsf{S}})$ if and only if*

$$\exists d_2, d_3.\, d' \Rightarrow_S d_2 \wedge d_2 \xrightarrow{\overline{a(f_a^{\mathsf{M}}(d,e))}}_S d_3 \wedge (d, d_2) \in \eta(X^{\mathsf{M},\mathsf{S}}) \wedge (g_a^{\mathsf{M}}(d, e), d_3) \in \eta(X^{\mathsf{M},\mathsf{S}})$$

Proof. We drop the superscripts M, S when no confusion arises. We define sets $\mathcal{R}_i^{a,d,e} \subseteq D^{\mathsf{S}}$, for any $a \in Act$, d, e, $i \geq 0$, and depending on $\eta(X)$, as follows:

$$\begin{cases} \mathcal{R}_0^{a,d,e} = \{d' \mid \exists d_3.\, d' \xrightarrow{\overline{a(f_a^{\mathsf{M}}(d,e))}}_S d_3 \wedge (d, d') \in \eta(X) \wedge (g_a^{\mathsf{M}}(d, e), d_3) \in \eta(X)\} \\ \mathcal{R}_{i+1}^{a,d,e} = \{d' \mid \exists d_2.\, d' \xrightarrow{\tau}_S d_2 \wedge d_2 \in R_i^{a,d,e}\} \end{cases}$$

And let $\mathcal{R}^{a,d,e} = \bigcup_{i \geq 0} \mathcal{R}_i^{a,d,e}$. Obviously, by definition of $\Rightarrow$, we have

$$\begin{aligned} \mathcal{R}^{a,d,e} = \{d' \mid \exists d_2, d_3.\, d' \Rightarrow_S d_2 \wedge d_2 \xrightarrow{\overline{a(f_a^{\mathsf{M}}(d,e))}}_S d_3 \wedge (d, d_2) \in \eta(X) \\ \wedge (g_a^{\mathsf{M}}(d, e), d_3) \in \eta(X)\} \end{aligned}$$

We will prove, using an approximation method, that this coincides with the minimal solution of $Y_a^{\mathsf{M},\mathsf{S}}$. More precisely, we claim:

$$((d, d', e) \in \theta(Y_a^{\mathsf{M},\mathsf{S}})) = (d' \in \mathcal{R}^{a,d,e})$$

Recall that according to the algorithm, Y_a is of the form

$$Y_a(d, d', e) = (X(d, d') \wedge \Xi) \vee \exists e'_\tau.(h_\tau^{\mathsf{S}}(d', e'_\tau) \wedge Y_a(d, g_\tau^{\mathsf{S}}(d', e'_\tau), e)) \qquad (1)$$

where Ξ (generated by function $step$) is of the form

$$\begin{aligned} &(a = \tau \wedge X(g_\tau^{\mathsf{M}}(d, e), d')) \vee \\ &\exists e'_a.h_a^{\mathsf{S}}(d', e'_a) \wedge (f_a^{\mathsf{M}}(d, e) = f_a^{\mathsf{S}}(d', e'_a)) \wedge X(g_a^{\mathsf{M}}(d, e), g_a^{\mathsf{S}}(d', e'_a)) \end{aligned}$$

Note that, using the operational semantics for LPE S,

$$[\![X(d, d') \wedge \Xi]\!]\eta = \exists d''.\, (d, d') \in \eta(X) \wedge (g_a^{\mathsf{M}}(d, e), d'') \in \eta(X) \wedge d' \xrightarrow{\overline{a(f_a^{\mathsf{M}}(d,e))}}_S d''$$

Hence,

$$[\![X(d, d') \wedge \Xi]\!]\eta = (d' \in \mathcal{R}_0^{a,d,e}) \qquad (2)$$

We next show by induction on n, that the finite approximations $Y_a^n(d, d', e)$ of equation (1) can be characterized by the following equation:

$$Y_a^n(d, d', e) = (d' \in \bigcup_{0 \le i < n} \mathcal{R}_i^{a,d,e})$$

The basis is trivial ($Y_a = \emptyset$). For the induction step, it suffices to note that

$$\begin{aligned}
&\{d' \mid Y_a^{n+1}(d, d', e)\} \\
&\stackrel{*}{=} \{d' \mid ((d, d') \in \eta(X) \wedge [\![\Xi]\!]\eta) \vee \exists e'_\tau.(h_\tau^{\mathsf{S}}(d', e'_\tau) \wedge g_\tau^{\mathsf{S}}(d', e'_\tau) \in \bigcup_{0 \le i < n} \mathcal{R}_i^{a,d,e})\} \\
&= \{d' \mid (d, d') \in \eta(X) \wedge [\![\Xi]\!]\eta\} \cup \bigcup_{0 \le i < n} \{d' \mid \exists e'_\tau.(h_\tau^{\mathsf{S}}(d', e'_\tau) \wedge g_\tau^{\mathsf{S}}(d', e'_\tau) \in \mathcal{R}_i^{a,d,e})\} \\
&\stackrel{\bigstar}{=} \mathcal{R}_0^{a,d,e} \cup \bigcup_{0 \le i < n} \mathcal{R}_{i+1}^{a,d,e} \\
&= \bigcup_{0 \le i < n+1} \mathcal{R}_i^{a,d,e},
\end{aligned}$$

where the step $(*)$ uses the induction hypothesis, and the step ($\bigstar$) uses equation (2) above, and the definition of $\mathcal{R}_i^{a,d,e}$.

Next we compute the first infinitary approximation Y_a^ω of equation (1):

$$\begin{aligned}
\{d' \mid Y_a^\omega(d, d', e)\} &= \bigcup_{n \ge 0} \{d' \mid Y_a^n(d, d', e)\} \\
&= \bigcup_{n \ge 0} \bigcup_{0 \le i < n} \mathcal{R}_i^{a,d,e} \\
&= \bigcup_{i \ge 0} \mathcal{R}_i^{a,d,e}
\end{aligned}$$

It remains to show that the solution is stable, i.e. Y^ω is a solution of equation (1). This can be readily checked as follows:

$$\begin{aligned}
&\{d' \mid ((d, d') \in \eta(X) \wedge [\![\Xi]\!]\eta) \vee \exists e'_\tau.(h_\tau^{\mathsf{S}}(d', e'_\tau) \wedge g_\tau^{\mathsf{M}}(d', e'_\tau) \in \mathcal{R}^{a,d,e})\} \\
&= \mathcal{R}_0 \cup \bigcup_{i \ge 1} \mathcal{R}_i^{a,d,e} \\
&= \mathcal{R}^{a,d,e}
\end{aligned}$$

Hence we have found the correct minimal solution of μE_1. □

Finally, the correctness of the algorithm follows from the following theorem.

Theorem 2. *Let $\nu E_2 \mu E_1$ be the equation system generated by Algorithm 1 on M and S and $\theta = [\![\nu E_2 \mu E_1]\!]$. Then for all d and d' we have $M(d) \underline{\leftrightarrow}_b S(d')$ if and only if $(d, d') \in \theta(X^{\mathsf{M},\mathsf{S}})$.*

Proof. Recall that according to the algorithm, $X^{\mathsf{M},\mathsf{S}}$ is of the form

$$X^{\mathsf{M},\mathsf{S}}(d,d') = \bigwedge_{a\in Act} \forall e_a.(h_a^{\mathsf{M}}(d,e_a) \implies Y_a^{\mathsf{M},\mathsf{S}}(d,d',e_a)) \\ \wedge \bigwedge_{a\in Act} \forall e'_a.(h_a^{\mathsf{S}}(d',e'_a) \implies Y_a^{\mathsf{S},\mathsf{M}}(d',d,e'_a))$$

By symmetry, w.l.o.g. we only consider $\bigwedge_{a\in Act} \forall e_a.(h_a^{\mathsf{M}}(d,e_a) \implies Y_a^{\mathsf{M},\mathsf{S}}(d,d',e_a))$.

We define $G : D^{\mathsf{M}} \times D^{\mathsf{S}} \to D^{\mathsf{M}} \times D^{\mathsf{S}}$ as

$$G(\mathcal{B}) = \{(d,d') \mid \bigwedge_{a\in Act} \forall e_a.(h_a^{\mathsf{M}}(d,e_a) \implies (d,d',e_a) \in \eta(Y_a^{\mathsf{M},\mathsf{S}}))\}$$

where $\eta = [\![\mu E_1]\!][\mathcal{B}/X^{\mathsf{M},\mathsf{S}}]$.

Note that by [17, Lemma 5], G is monotonic, and thus the maximal fixpoint of G exists which is denoted by $\nu B.G(B)$. According to the semantics of PBES (cf. Definition 8), we have

$$\nu B.G(B) = \{(d,d') \mid (d,d') \in \theta(X^{\mathsf{M},\mathsf{S}})\}$$

Recall that the functional $\mathcal{F}$ is defined as

$$\mathcal{F}(\mathcal{B}) = \{(d,d') \mid \forall a \in Act, e_a \in E_a.h_a^{\mathsf{M}}(d,e_a) \implies \\ \exists d_2,d_3.d' \Rightarrow d_2 \wedge d_2 \xrightarrow{\overline{a(f_a^{\mathsf{M}}(d,e_a))}}_S d_3 \wedge (d,d_2) \in \mathcal{B} \wedge (g_a^{\mathsf{M}}(d,e_a),d_3) \in \mathcal{B}\}$$

We claim that for any $\mathcal{B}$,

$$\mathcal{F}(\mathcal{B}) = G(\mathcal{B})$$

To see this, first let us note that by Theorem 1

$$\eta(Y_a^{\mathsf{M},\mathsf{S}}) = \{d' \mid \exists d_2,d_3.\, d' \Rightarrow d_2 \wedge d_2 \xrightarrow{\overline{a(f_a^{\mathsf{M}}(d,e))}}_S d_3 \wedge \mathcal{B}(d,d_2) \wedge \mathcal{B}(g_a^{\mathsf{M}}(d,e),d_3)\}$$

It follows that

$$\begin{aligned} &G(\mathcal{B}) \\ &= \{(d,d') \mid \bigwedge_{a\in Act} \forall e_a.(h_a(d,e_a) \implies (d,d',e_a) \in \eta(Y_a^{\mathsf{M},\mathsf{S}})\} \\ &= \{(d,d') \mid \bigwedge_{a\in Act} \forall e_a.(h_a(d,e_a) \implies \exists d_2,d_3.\, d' \Rightarrow_S d_2 \wedge d_2 \xrightarrow{\overline{a(f_a^{\mathsf{M}}(d,e))}}_S d_3 \wedge \\ &\qquad\qquad \mathcal{B}(d,d_2) \wedge \mathcal{B}(g_a^{\mathsf{M}}(d,e),d_3)\} \\ &= \mathcal{F}(\mathcal{B}) \end{aligned}$$

It follows from Lemma 1 that

$$\underline{\leftrightarrow}_b = \nu\mathcal{F} = \nu B.G(B) = \{(d,d') \mid (d,d') \in \theta(X^{\mathsf{M},\mathsf{S}})\}$$

from which it is not difficult to see that $(d,d') \in \theta(X)$ if and only if $M(d) \underline{\leftrightarrow}_b S(d')$.

□

5 Example: Unbounded Queues

In this section we demonstrate the potential of the technique outlined in the previous section by applying it to an example of unbounded queues. The capacity of a bounded queue is doubled by connecting a queue of the same capacity. This means that a composition of bounded queues is behaviorally different from the constituent queues. In contrast, a composition of queues with infinite capacity does not change the behavior, as this again yields an unbounded queue.

Let D be an arbitrary data sort (possibly infinite sized) which is equipped with an equality relation, and let $\mathcal{Q}$ denote the data sort of queues of infinite capacity. We denote the empty queue by $[]$ and for any $d \in D$ we denote the queue containing only d by $[d]$. Operations on queues include $q{+\!\!+}q'$, denoting the natural concatenation of queues q and q', and functions $hd : \mathcal{Q} \to D$ and $tl : \mathcal{Q} \to \mathcal{Q}$ which yield the head and tail of a queue q, respectively.

The processes S and T defined below model the composition of two unbounded queues and three unbounded queues, respectively. Remark that we obtained LPEs S and T as a result of an automated linearization of the parallel composition of two (resp. three) queues of infinite capacity. These original specifications have been omitted for brevity. Processes S and T can communicate with their environments via parameterized actions $r(d)$ (read d from the environment) and $w(d)$ (write d to the environment). The τ actions represent the internal communication of data from one queue to the next.

$$
\begin{array}{ll}
S(s_0, s_1 : \mathcal{Q}) = & T(t_0, t_1, t_2 : \mathcal{Q}) = \\
\quad \sum_{v:D} r(v) \cdot S([v]{+\!\!+}s_0, s_1) & \quad \sum_{u:D} r(u) \cdot T([u]{+\!\!+}t_0, t_1, t_2) \\
+ s_1 \neq [] \implies w(hd(s_1)) \cdot S(s_0, tl(s_1)) & + t_2 \neq [] \implies w(hd(t_2)) \cdot T(t_0, t_1, tl(t_2)) \\
+ s_0 \neq [] \implies \tau \cdot S(tl(s_0), [hd(s_0)]{+\!\!+}s_1) & + t_0 \neq [] \implies \tau \cdot T(tl(t_0), [hd(t_0)]{+\!\!+}t_1, t_2) \\
 & + t_1 \neq [] \implies \tau \cdot T(t_0, tl(t_1), [hd(t_1)]{+\!\!+}t_2)
\end{array}
$$

Applying Algorithm 1 for processes S and T, we obtain a PBES consisting of 8 equations. For lack of space, only the two most interesting fragments of the PBES are shown below.

$$
\begin{array}{l}
\big(\nu X^{S,T}(s_0, s_1, t_0, t_1, t_2 : \mathcal{Q}) = \ldots \wedge \ (s_1 \neq [] \implies Y_w^{S,T}(s_0, s_1, t_0, t_1, t_2)) \ \wedge \ldots\big) \\
\vdots \\
\big(\mu Y_w^{S,T}(s_0, s_1, t_0, t_1, t_2 : \mathcal{Q}) = \ (t_0 \neq [] \wedge Y_w^{S,T}(s_0, s_1, tl(t_0), [hd(t_0)]{+\!\!+}t_1, t_2)) \vee \\
\quad (t_1 \neq [] \wedge Y_w^{S,T}(s_0, s_1, t_0, tl(t_1), [hd(t_1)]{+\!\!+}t_2)) \vee (t_2 \neq [] \wedge hd(t_2) = hd(s_1) \wedge \\
\quad X^{S,T}(s_0, s_1, t_0, t_1, t_2) \wedge X^{S,T}(s_0, tl(s_1), t_0, t_1, tl(t_2))) \big) \\
\vdots
\end{array}
$$

In the remainder of this section, we strongly rely on techniques for solving and manipulating PBESs like adding invariants, symbolic approximations and strengthening equations. Some of these techniques have already been automated (e.g. symbolic approximation, see [16]). For a detailed account of all techniques, we refer to [16,17].

Consider the equation for $Y_w^{S,T}$. It represents the case where process T has to simulate a $w(hd(s_1))$ action of process S by possibly executing a finite number of τ-steps before executing action $w(hd(t_2))$. Inspired by the scenario that captures the minimal amount of τ-steps that are needed (two steps when $t_1 = t_2 = []$, one when $t_2 = [] \neq t_1$ and none otherwise), we strengthen the equation for $Y_w^{S,T}$ as follows:

$$
\begin{aligned}
&\mu Y_w^{S,T}(s_0,s_1,t_0,t_1,t_2:\mathcal{Q}) = \\
&\quad (t_0 \neq [] \wedge \underline{t_1 = t_2 = []} \wedge Y_w^{S,T}(s_0,s_1,tl(t_0),[hd(t_0)] +\!\!+ t_1,t_2)) \vee \\
&\quad (t_1 \neq [] \wedge \underline{t_2 = []} \wedge Y_w^{S,T}(s_0,s_1,t_0,tl(t_1),[hd(t_1)] +\!\!+ t_2)) \vee \\
&\quad (t_2 \neq [] \wedge hd(t_2) = hd(s_1) \wedge X^{S,T}(s_0,s_1,t_0,t_1,t_2) \wedge X^{S,T}(s_0,tl(s_1),t_0,t_1,tl(t_2)))
\end{aligned}
$$

The solution to $Y_w^{S,T}$ can be found by a straightforward symbolic approximation. This stabilizes at the fourth approximation, and can — depending on the rewriting technology that is used — be found automatically. The resulting solution is:

$$
\begin{aligned}
&\mu Y_w^{S,T}(s_0,s_1,t_0,t_1,t_2:\mathcal{Q}) = \\
&\quad (t_0 \neq [] \wedge t_1 = t_2 = [] \wedge hd(t_0) = hd(s_1) \\
&\quad \wedge X^{S,T}(s_0,s_1,tl(t_0),[],[hd(t_0)]) \wedge X(s_0,tl(s_1),tl(t_0),[],[])) \vee \\
&\quad (t_1 \neq [] \wedge t_2 = [] \wedge hd(t_1) = hd(s_1) \wedge X^{S,T}(s_0,s_1,t_0,tl(t_1),[hd(t_1)]) \\
&\quad \wedge X^{S,T}(s_0,tl(s_1),t_0,tl(t_1),[])) \vee \\
&\quad (t_2 \neq [] \wedge hd(t_2) = hd(s_1) \wedge X^{S,T}(s_0,s_1,t_0,t_1,t_2) \wedge X^{S,T}(s_0,tl(s_1),t_0,t_1,tl(t_2)))
\end{aligned}
$$

The solution to the (omitted) equation $Y_w^{T,S}$ can be obtained analogously. Likewise, we can strengthen and subsequently solve the equations for the Y_τ's and the Y_r's. The resulting solutions can be substituted in the equation for $X^{S,T}$ yielding the following closed equation for $X^{S,T}$.

$$
\begin{aligned}
&\nu X^{S,T}(s_0,s_1,t_0,t_1,t_2:\mathcal{Q}) = \\
&\quad X^{S,T}(s_0,s_1,t_0,t_1,t_2) \wedge (\forall v:D\,.\,X^{S,T}([v] +\!\!+ s_0,s_1,[v] +\!\!+ t_0,t_1,t_2)) \\
&\wedge (s_1 \neq [] \implies ((t_0 \neq [] \wedge t_1 = [] \wedge t_2 = [] \wedge hd(t_0) = hd(s_1) \wedge \\
&\qquad X^{S,T}(s_0,s_1,tl(t_0),[],[hd(t_0)]) \wedge X^{S,T}(s_0,tl(s_1),tl(t_0),[],[])) \\
&\qquad \vee (t_1 \neq [] \wedge t_2 = [] \wedge hd(t_1) = hd(s_1) \wedge \\
&\qquad\quad X^{S,T}(s_0,s_1,t_0,tl(t_1),[hd(t_1)]) \wedge X^{S,T}(s_0,tl(s_1),t_0,tl(t_1),[])) \\
&\qquad \vee (t_2 \neq [] \wedge hd(t_2) = hd(s_1) \wedge X^{S,T}(s_0,s_1,t_0,t_1,t_2) \wedge \\
&\qquad\quad X^{S,T}(s_0,tl(s_1),t_0,t_1,tl(t_2))))) \\
&\wedge (s_0 \neq [] \implies (X^{S,T}(s_0,s_1,t_0,t_1,t_2) \wedge (X^{S,T}(tl(s_0),[hd(s_0)] +\!\!+ s_1,t_0,t_1,t_2) \\
&\qquad \vee (t_0 \neq [] \wedge X^{S,T}(tl(s_0),[hd(s_0)] +\!\!+ s_1,tl(t_0),[hd(t_0)] +\!\!+ t_1,t_2)) \\
&\qquad \vee (t_1 \neq [] \wedge X^{S,T}(tl(s_0),[hd(s_0)] +\!\!+ s_1,t_0,tl(t_1),[hd(t_1)] +\!\!+ t_2))))) \\
&\wedge (t_2 \neq [] \implies ((s_0 \neq [] \wedge s_1 = [] \wedge hd(s_0) = hd(t_2) \wedge \\
&\qquad X^{S,T}(tl(s_0),[hd(s_0)],t_0,t_1,t_2) \wedge X^{S,T}(tl(s_0),[],t_0,t_1,tl(t_2))) \\
&\qquad \vee (s_1 \neq [] \wedge hd(s_1) = hd(t_2) \wedge X^{S,T}(s_0,s_1,t_0,t_1,t_2) \wedge \\
&\qquad\quad X^{S,T}(s_0,tl(s_1),t_0,t_1,tl(t_2))))) \\
&\\
&\wedge ((t_0 \neq [] \vee t_1 \neq []) \implies (X^{S,T}(s_0,s_1,t_0,t_1,t_2) \wedge \\
&\qquad (X^{S,T}(s_0,s_1,tl(t_0),[hd(t_0)] +\!\!+ t_1,t_2) \vee X^{S,T}(s_0,s_1,t_0,tl(t_1),[hd(t_1)] +\!\!+ t_2) \vee \\
&\qquad X^{S,T}(tl(s_0),[hd(s_0)] +\!\!+ s_1,t_0,tl(t_1),[hd(t_1)] +\!\!+ t_2) \vee \\
&\qquad (s_0 \neq [] \wedge (X^{S,T}(tl(s_0),[hd(s_0)] +\!\!+ s_1,tl(t_0),[hd(t_0)] +\!\!+ t_1,t_2))))))
\end{aligned}
$$

Utilizing the fact that $s_0 \mathbin{+\!\!+} s_1 = t_0 \mathbin{+\!\!+} t_1 \mathbin{+\!\!+} t_2$ is an invariant of the closed equation $X^{S,T}$, the symbolic approximation of $X^{S,T}$ stabilizes at the third approximation, yielding the solution $s_0 \mathbin{+\!\!+} s_1 = t_0 \mathbin{+\!\!+} t_1 \mathbin{+\!\!+} t_2$ [1]. Evaluating the solution to $X^{S,T}$ for the initial values $s_0 = s_1 = t_0 = t_1 = t_2 = []$ tells us that $S([],[])$ and $T([],[],[])$ are branching bisimilar. In fact, all processes $S(s_0, s_1)$ and $T(t_0, t_1, t_2)$ satisfying the condition $s_0 \mathbin{+\!\!+} s_1 = t_0 \mathbin{+\!\!+} t_1 \mathbin{+\!\!+} t_2$ are branching bisimilar.

6 Transformation for Other Equivalences

In this section, we demonstrate how we can adapt the algorithm presented in Section 4 to other variants of bisimulation. The strong case is simple and somehow known in [19] modulo different formalisms. The algorithm is included in [7]. As discussed in the introduction, our encoding for weak bisimulation (see Algorithm 2) fixes the generally incorrect encoding found in [18]. The case for (branching) simulation equivalence (see Algorithm 3) is novel. The correctness proofs are similar to the case for branching bisimulation.

Algorithm 2. Generation of a PBES for Weak Bisimulation

$wbisim = \nu E_2 \mu E_1$, **where**

$$E_2 := \{X^{\mathsf{M},\mathsf{S}}(d : D^{\mathsf{M}}, d' : D^{\mathsf{S}}) = match^{\mathsf{M},\mathsf{S}}(d, d') \wedge match^{\mathsf{S},\mathsf{M}}(d', d)\ ,\\ X^{\mathsf{S},\mathsf{M}}(d' : D^{\mathsf{S}}, d : D^{\mathsf{M}}) = X^{\mathsf{M},\mathsf{S}}(d, d')\ \}$$

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

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

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

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

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

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

7 Conclusion

We have shown how to transform the weak and branching (bi)simulation equivalence checking problems for infinite systems to solving Parameterized Boolean Equation Systems. We demonstrated our method on a small example, showing that the concatenation of two unbounded queues is branching bisimilar to the concatenation of three

[1] Remark that the fact that the solution and the invariant match is coincidental: it is clear that e.g. the trivial invariant *true* ($\top$) does not exhibit this phenomenon.

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

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

$$E_2 := \{X(d : D^{\mathsf{M}}, d' : D^{\mathsf{S}}) = X^{\mathsf{M},\mathsf{S}}(d,d') \wedge X^{\mathsf{S},\mathsf{M}}(d',d),$$
$$X^{\mathsf{M},\mathsf{S}}(d : D^{\mathsf{M}}, d' : D^{\mathsf{S}}) = match^{\mathsf{M},\mathsf{S}}(d,d'),$$
$$X^{\mathsf{S},\mathsf{M}}(d' : D^{\mathsf{S}}, d : D^{\mathsf{M}}) = match^{\mathsf{S},\mathsf{M}}(d',d)\}$$
$$E_1 := \{Y_a^{\mathsf{p},\mathsf{q}}(m,n,e) = close_a^{\mathsf{p},\mathsf{q}}(d,d',e) \mid a \in Act\}$$

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

$$match^{\mathsf{p},\mathsf{q}}(d : D^{\mathsf{p}}, d' : D^{\mathsf{q}}) = \bigwedge_{a \in Act} \forall e : E_a^{\mathsf{p}}.\, (h_a^{\mathsf{p}}(d,e) \implies Y_a^{\mathsf{p},\mathsf{q}}(d,d',e));$$

$$close_a^{\mathsf{p},\mathsf{q}}(d : D^{\mathsf{p}}, d' : D^{\mathsf{q}}, e : E_a^{\mathsf{p}}) = \exists e' : E_\tau^{\mathsf{q}}.\, (h_\tau^{\mathsf{q}}(d',e') \wedge Y_a^{\mathsf{p},\mathsf{q}}(d, g_\tau^{\mathsf{q}}(d',e'),e))$$
$$\vee (X^{\mathsf{p},\mathsf{q}}(d,d') \wedge step_a^{\mathsf{p},\mathsf{q}}(d,d',e));$$

$$step_a^{\mathsf{p},\mathsf{q}}(d : D^{\mathsf{p}}, d' : D^{\mathsf{q}}, e : E_a^{\mathsf{p}}) = (a = \tau \wedge X^{\mathsf{p},\mathsf{q}}(g_\tau^{\mathsf{p}}(d,e),d')) \vee$$
$$\exists e' : E_a^{\mathsf{q}}.\, h_a^{\mathsf{q}}(d',e') \wedge (f_a^{\mathsf{p}}(d,e) = f_a^{\mathsf{q}}(d',e')) \wedge X^{\mathsf{p},\mathsf{q}}(g_a^{\mathsf{p}}(d,e), g_a^{\mathsf{q}}(d',e'));$$

unbounded queues. This example could not be solved directly with the cones and foci method (without introducing a third process), because these systems are not functionally branching bisimilar, and moreover, both systems perform τ-steps.

Our solution is a symbolic verification algorithm. Compared with the previously known algorithms, it has the advantage that the solution of the PBES indicates exactly which states of the implementation and specification are bisimilar. This provides some positive feedback in case the initial states of the two systems are not bisimilar. Note that we have introduced a *generic* scheme that can be applied to other weak equivalences and preorders in branching time spectrum [10], and also to other formalisms of concurrency.

We conjecture that for infinite systems, it is essential that the PBES has alternation depth two, as opposed to the finite case. We leave it for future work to apply our method to various equivalences for mobile processes, in particular π-calculus [25], such as weak early, late and open bisimulation. Orthogonal to this, we shall continue our work on improving tool support for solving PBESs, and the application of our techniques to larger specifications of infinite systems.

Acknowledgments. We are grateful to Wan Fokkink and Jan Friso Groote for stimulating discussions.

References

1. Alur, R., Dill, D.L.: A theory of timed automata. Theoretical Computer Science 126(2), 183–235 (1994)
2. Andersen, H.R.: Model checking and boolean graphs. Theoretical Computer Science 126(1), 3–30 (1994)
3. Andersen, H.R., Vergauwen, B.: Efficient checking of behavioural relations and modal assertions using fixed-point inversion. In: Wolper, P. (ed.) CAV 1995. LNCS, vol. 939, pp. 142–154. Springer, Heidelberg (1995)

4. Basten, T.: Branching bisimilarity is an equivalence indeed! Information Processing Letters 58, 141–147 (1996)
5. Bolognesi, T., Brinksma, E.: Introduction to the ISO specification language LOTOS. Computer Networks 14, 25–59 (1987)
6. Chandy, K.M., Misra, J.: Parallel Program Design: A Foundation. Addison-Wesley, Reading (1988)
7. Chen, T., Ploeger, B., van de Pol, J., Willemse, T.A.C.: Equivalence checking for infinite systems using parameterized boolean equation systems. CS-Report 07-14, Technische Universiteit Eindhoven (2007)
8. Cleaveland, R., Steffen, B.: Computing behavioural relations, logically. In: Leach Albert, J., Monien, B., Rodríguez-Artalejo, M. (eds.) Automata, Languages and Programming. LNCS, vol. 510, pp. 127–138. Springer, Heidelberg (1991)
9. Fokkink, W., Pang, J., van de Pol, J.: Cones and foci: A mechanical framework for protocol verification. Formal Methods in System Design 29(1), 1–31 (2006)
10. van Glabbeek, R.: The Linear Time - Branching Time Spectrum II. In: Best, E. (ed.) CONCUR 1993. LNCS, vol. 715, pp. 66–81. Springer, Heidelberg (1993)
11. van Glabbeek, R.J., Weijland, W.P.: Branching time and abstraction in bisimulation semantics. Journal of the ACM 43, 555–600 (1996)
12. Groote, J.F., Mateescu, R.: Verification of temporal properties of processes in a setting with data. In: Haeberer, A.M. (ed.) AMAST 1998. LNCS, vol. 1548, pp. 74–90. Springer, Heidelberg (1998)
13. Groote, J.F., van de Pol, J.: A bounded retransmission protocol for large data packets. In: Nivat, M., Wirsing, M. (eds.) AMAST 1996. LNCS, vol. 1101, pp. 536–550. Springer, Heidelberg (1996)
14. Groote, J.F., Reniers, M.: Algebraic process verification. In: Bergstra, J.A., Ponse, A., Smolka, S.A. (eds.) Handbook of Process Algebra, pp. 1151–1208. Elsevier, Amsterdam (2001)
15. Groote, J.F., Vaandrager, F.W.: An efficient algorithm for branching bisimulation and stuttering equivalence. In: Paterson, M.S. (ed.) Automata, Languages and Programming. LNCS, vol. 443, pp. 626–638. Springer, Heidelberg (1990)
16. Groote, J.F., Willemse, T.A.C.: Model-checking processes with data. Science of Computer Programming 56(3), 251–273 (2005)
17. Groote, J.F., Willemse, T.A.C.: Parameterised boolean equation systems. Theoretical Computer Science 343(3), 332–369 (2005)
18. Kwak, H., Choi, J., Lee, I., Philippou, A.: Symbolic weak bisimulation for value-passing calculi. Technical Report, MS-CIS-98-22, Department of Computer and Information Science, University of Pennsylvania (1998)
19. Lin, H.: Symbolic transition graph with assignment. In: Sassone, V., Montanari, U. (eds.) CONCUR 1996. LNCS, vol. 1119, pp. 50–65. Springer, Heidelberg (1996)
20. Lynch, N., Tuttle, M.: An introduction to input/output automata. CWI Quarterly 2(3), 219–246 (1989)
21. Mader, A.: Verification of modal properties using boolean equation systems. PhD Thesis, VERSAL 8, Bertz Verlag, Berlin (1997)
22. Mateescu, R.: A generic on-the-fly solver for alternation-free boolean equation systems. In: Garavel, H., Hatcliff, J. (eds.) ETAPS 2003 and TACAS 2003. LNCS, vol. 2619, pp. 81–96. Springer, Heidelberg (2003)
23. Milner, R.: A Calculus of Communicating Systems. Springer, Heidelberg (1980)
24. Milner, R.: Communication and Concurrency. Prentice-Hall, Englewood Cliffs (1989)

25. Milner, R., Parrow, J., Walker, D.: A calculus of mobile processes (Part I/II). Information and Computation 100(1), 1–77 (1992)
26. Paige, R., Tarjan, R.: Three partition refinement algorithms. SIAM Journal of Computing 16(6), 973–989 (1987)
27. Tarski, A.: A lattice-theoretical fixpoint theorem and its applications. Pacific Journal of Mathematics 5(2), 285–309 (1955)
28. Zhang, D., Cleaveland, R.: Fast generic model-checking for data-based systems. In: Wang, F. (ed.) FORTE 2005. LNCS, vol. 3731, pp. 83–97. Springer, Heidelberg (2005)