Reduced Dependency Spaces for Existential Parameterised Boolean Equation Systems[*]

Yutaro Nagae
Graduate School of Information Science
Nagoya University
nagae_y@trs.cm.is.nagoya-u.ac.jp

Masahiko Sakai
Graduate School of Informatics
Nagoya University
sakai@i.nagoya-u.ac.jp

A parameterised Boolean equation system (PBES) is a set of equations that defines sets satisfying the equations as the least and/or greatest fixed-points. Thus this system is regarded as a declarative program defining predicates, where a program execution returns whether a given ground atomic formula holds or not. The program execution corresponds to the membership problem of PBESs, which is however undecidable in general.

This paper proposes a subclass of PBESs which expresses universal-quantifiers free formulas, and studies a technique to solve the problem on it. We use the fact that the membership problem is reduced to the problem whether a proof graph exists. To check the latter problem, we introduce a so-called dependency space which is a graph containing all of the minimal proof graphs. Dependency spaces are, however, infinite in general. Thus, we propose some conditions for equivalence relations to preserve the result of the membership problem, then we identify two vertices as the same under the relation. In this sense, dependency spaces possibly result in a finite graph. We show some examples having infinite dependency spaces which are reducible to finite graphs by equivalence relations. We provide a procedure to construct finite dependency spaces and show the soundness of the procedure. We also implement the procedure using an SMT solver and experiment on some examples including a downsized McCarthy 91 function.

1 Introduction

A *Parameterised Boolean Equation System* (PBES) [13, 10, 12] is a set of equations denoting some sets as the least and/or greatest fixed-points. PBESs can be used as a powerful tool for solving a variety of problems such as process equivalences [1], model checking [13, 11], and so on.

We explain PBESs by an example PBES $\mathcal{E}_1$, which consists of the following two equations:

$$\begin{aligned} \nu X(n:N) &= X(n+1) \vee Y(n) \\ \mu Y(n:N) &= Y(n+1) \end{aligned}$$

$X(n:N)$ denotes that n is a natural number and a formal parameter of X. Each of the predicate variables X and Y represents a set of natural numbers regarding that $X(n)$ is true if and only if n is in X. These sets are determined by the equations, where μ (resp. ν) is a least (resp. greatest) fixed-point operator. In the PBES $\mathcal{E}_1$, Y is an empty set since Y is the least set satisfying that $Y(n)$ iff $Y(n+1)$ for any $n \geq 0$. Similarly, X is equal to $\mathbb{N}$ since X is the greatest set satisfying that $X(n)$ iff $X(n+1) \vee Y(n)$ for any $n \geq 0$. A PBES is regarded as a declarative program defining predicates. In this example, an execution of the program for an input $X(0)$ outputs true.

The membership problem for PBESs is undecidable in general [13]. Undecidability is proved by a reduction of the model checking problem for the modal μ-calculus with data. Some techniques have

[*]This paper is partially supported by JSPS KAKENHI Grant Number JP17H01721.

been proposed to solve the problem for some subclasses of PBESs: one by instantiating a PBES to a *Boolean Equation System* (BES) [17], one by calculating invariants [16], and one by constructing a proof graph [4]. In the last method, the membership problem is reduced to an existence of a proof graph. If there exists a finite proof graph for a given instance of the problem, it is not difficult to find it mechanically. However, finite proof graphs do not always exist. A technique is proposed in [15] that possibly produces a finite reduced proof graph, which represents an infinite proof graph. The technique manages the disjunctive PBESs, in which data-quantifiers are not allowed.

In this paper, we propose a more general subclass, named *existential PBESs*, and extend the notion of dependency spaces. We discuss the relation between extended dependency spaces and the existence of proof graphs. Dependency spaces are, however, infinite graphs in most cases. Thus we reduce a dependency space for the existential class to a finite one in a more sophisticated way based on the existing technique in [15]. We also give a procedure to construct a reduced dependency space and show the soundness of the procedure. We explain its implementation and an experiment on some examples.

2 PBESs and Proof Graphs

We follow [4] and [12] for basic notions related to PBESs and proof graphs.

We assume a set $\mathcal{DS}$ of *data sorts*. For every data sort $D \in \mathcal{DS}$, we assume a set $\mathcal{V}_D$ of *data variables* and a *semantic domain* $\mathbb{D}$ corresponding to it. In this paper, we assume $B, N \in \mathcal{DS}$ corresponding to the Boolean domain $\mathbb{B} = \{\mathbb{t}, \mathbb{f}\}$ and the natural numbers $\mathbb{N}$, respectively. We use D to represent a sort in $\mathcal{DS}$, $\mathbb{D}$ for the semantic domain corresponding to D, and d and e as a data variable in $\mathcal{V}_D$. We assume appropriate *data functions* according to operators, and use $[[exp]]\delta$ to represent a value obtained by the evaluation of a *data expression exp* under a data environment δ. A data expression interpreted to a value in $\mathbb{B}$ is called a *Boolean expression*. We write $\mathbf{a}$ or $\overrightarrow{a}$ by using boldfaced font or an arrow to represent a sequence $a_1, \ldots, a_n$ of objects. Especially, $\mathbf{d}{:}\mathbf{D}$ is an abbreviation of a sequence $d_1{:}D_1, \ldots, d_n{:}D_n$. We write $\mathbb{D}^*$ as a product $\mathbb{D}_1 \times \cdots \times \mathbb{D}_n$ of appropriate domains. In this paper, we use usual operators and constants like true, false, $\leq$, 0, 1, $+$, $-$, and so on, along with expected data functions.

A *Parameterised Boolean Equation System* (PBES) $\mathcal{E}$ is a sequence of well-sorted equations:

$$(\sigma_1 X_1(\mathbf{d}{:}\mathbf{D}) = \varphi_1) \cdots (\sigma_n X_n(\mathbf{d}{:}\mathbf{D}) = \varphi_n)$$

where φ_i is a *predicate formula* defined by the following BNF, and σ_i is either one of the quantifiers μ, ν used to indicate the least and greatest fixed-points, respectively ($1 \leq i \leq n$).

$$\varphi \quad ::= \quad b \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \forall d{:}D\ \varphi \mid \exists d{:}D\ \varphi \mid X(\mathbf{exp})$$

Here X is a predicate variable with fixed arity, b is a Boolean expression, d is a data variable in $\mathcal{V}_D$, and $\mathbf{exp}$ is a sequence of data expressions. We say $\mathcal{E}$ is *closed* if it does neither contain free predicate variables nor free data variables. Note that the negation is allowed only in expressions b or exp as a data function.

Example 1 *A PBES $\mathcal{E}_2$ is given as follows:*

$$\begin{array}{rcl} \nu X_1(n:N) & = & (n = 0 \wedge X_1(n+2)) \vee (n > 0 \wedge X_2(n-1) \wedge X_1(n+2)) \\ \mu X_2(n:N) & = & (n \geq 3 \wedge X_2(n-2)) \vee (n = 1 \wedge X_1(n-1)) \end{array}$$

Since the definition of the semantics is complex, we omit it and we will explain it by an example. The formal definition can be found in [12]. The meaning of a PBES is determined in the bottom-up order.

Considering a PBES $\mathcal{E}_2$ in Example 1, we first look at the second equation, which defines a set X_2. The set X_2 is fixed depending on the free variable X_1, i.e., the equation should be read as that X_2 is the least set satisfying the condition "$v \in X_2$ iff $(v \geq 3 \wedge v-2 \in X_2) \vee (v = 1 \wedge v-1 \in X_1)$" for any $v \in \mathbb{N}$. Thus the set X_2 is fixed as $\{1,3,5,\dots\}$ if $0 \in X_1$; $\emptyset$ otherwise, i.e., "$X_2(v)$ iff $\mathrm{odd}(v) \wedge X_1(0)$" for any $v \in \mathbb{N}$. Next, we replace the occurrence of X_2 in the first equation of $\mathcal{E}_2$ with $\mathrm{odd}(v) \wedge X_1(0)$, which results in "$\nu X_1(n : N) = (n = 0 \wedge X_1(n+2)) \vee (n > 0 \wedge \mathrm{odd}(n-1) \wedge X_1(0) \wedge X_1(n+2))$", if we simplify it. The set X_1 is fixed as the greatest set satisfying that $v \in X_1$ iff $(v = 0 \wedge v+2 \in X_1) \vee (v > 0 \wedge \mathrm{odd}(v-1) \wedge 0 \in X_1 \wedge v+2 \in X_1)$ for any $v \in \mathbb{N}$. All in all, we obtain $X_1 = \{0,2,4,\dots\}$ and $X_2 = \{1,3,5,\dots\}$. The solution $[\![\mathcal{E}]\!]$ of a closed PBES $\mathcal{E}$ is a function which takes a predicate variable, and returns a function on $\mathbb{D}^* \to \mathbb{B}$ that represents the corresponding predicate determined by the PBES. For instance, in the example PBES $\mathcal{E}_2$, $[\![\mathcal{E}_2]\!](X_1)$ is the function on $\mathbb{N} \to \mathbb{B}$ that returns $\mathbb{t}$ if and only if an even number is given, and $[\![\mathcal{E}_2]\!](X_2)$ is the function on $\mathbb{N} \to \mathbb{B}$ that returns $\mathbb{t}$ if and only if an odd number is given.

The *membership problem* for PBESs $\mathcal{E}$ is a problem that answers whether $X(\mathbf{v})$ holds (more formally $[\![\mathcal{E}]\!](X)(\mathbf{v}) = \mathbb{t}$) or not for a given predicate variable X and a value $\mathbf{v} \in \mathbb{D}^*$. The membership problem is characterized by proof graphs introduced in [4]. For a PBES $\mathcal{E} = (\sigma_1 X_1(\mathbf{d}{:}\mathbf{D}) = \varphi_1) \cdots (\sigma_n X_n(\mathbf{d}{:}\mathbf{D}) = \varphi_n)$, the *rank* of X_i $(1 \leq i \leq n)$ is the number of alternations of μ and ν in the sequence $\nu\sigma_1 \cdots \sigma_n$. Note that the rank of X_i bound with ν is even and the rank of X_i bound with μ is odd. For Example 1, $\mathrm{rank}_{\mathcal{E}_2}(X_1) = 0$ and $\mathrm{rank}_{\mathcal{E}_2}(X_2) = 1$. *Bound variables* are predicate variables X_i that occur in the left-hand sides of equations in $\mathcal{E}$. The set of bound variables is denoted by $\mathrm{bnd}(\mathcal{E})$. The *signature* $\mathrm{sig}(\mathcal{E})$ in $\mathcal{E}$ is defined by $\mathrm{sig}(\mathcal{E}) = \{(X_i, \mathbf{v}) \mid X_i \in \mathrm{bnd}(\mathcal{E}),\ \mathbf{v} \in \mathbb{D}^*\}$. We use $X_i(\mathbf{v})$ to represent $(X_i, \mathbf{v}) \in \mathrm{sig}(\mathcal{E})$. We use some graph theory terminology to introduce proof graphs. In a directed graph $\langle V, \to \rangle$, the *postset* of a vertex $v \in V$ is the set $\{v' \in V \mid v \to v'\}$.

Definition 2 *Let $\mathcal{E}$ be a PBES, $V \subseteq \mathrm{sig}(\mathcal{E})$, $\to \subseteq V \times V$, and $r \in \mathbb{B}$. The tuple $\langle V, \to, r \rangle$ is called a* proof graph *for the PBES if both of the following conditions hold:*

(1) For any $X_i(\mathbf{v}) \in V$, $\varphi_i(\mathbf{v})$ is evaluated to r under the assumption that the signatures in the postset of $X_i(\mathbf{v})$ are r and the other signatures are $\neg r$, where φ_i is the predicate formula that defines X_i.

(2) For any infinite sequence $Y_0(\mathbf{w_0}) \to Y_1(\mathbf{w_1}) \to \cdots$ in the graph, the minimum rank of Y^∞ is even, where Y^∞ is the set of Y_j that occurs infinitely often in the sequence.

We say that a proof graph $\langle V, \to, r \rangle$ *proves* $X_i(\mathbf{v}) = r$ if and only if $X_i(\mathbf{v}) \in V$. In the sequel, we consider the case that $r = \mathbb{t}$. The case $r = \mathbb{f}$ will derive dual results.

Example 3 *Consider the following graph with $r = \mathbb{t}$ and $\mathcal{E}_2$ in Example 1:*

$$\begin{tikzcd}
X_1(0) \arrow[rr] & & X_1(2) \arrow[rr] \arrow[dl] & & X_1(4) \arrow[r] \arrow[dl] & \dots \\
& X_2(1) \arrow[ul] & & X_2(3) \arrow[ll] & & \dots \arrow[ll]
\end{tikzcd}$$

This graph is a proof graph, which is justified from the following observations:

- *The graph satisfies the condition (1). For example, for a vertex $X_1(2)$, the predicate formula $\varphi_1(2) = (2 = 0 \wedge X_1(2+2)) \vee (2 > 0 \wedge X_2(2-1) \wedge X_1(2+2))$ is $\mathbb{t}$ assuming that $X_2(1) = X_1(4) = \mathbb{t}$.*
- *The graph satisfies the condition (2). For example, for an infinite sequence $X_1(0) \to X_1(2) \to X_2(1) \to X_1(0) \to \cdots$, the minimum rank of $\{X_1, X_2\}$ is 0.*

The next theorem states the relation between proof graphs and the membership problem on a PBES.

Theorem 4 ([4]) *For a PBES $\mathcal{E}$ and a $X_i(\mathbf{v}) \in \mathrm{sig}(\mathcal{E})$, the existence of a proof graph $\langle V, \to, r \rangle$ such that $X_i(\mathbf{v}) \in V$ coincides with $[\![\mathcal{E}]\!](X_i)(\mathbf{v}) = r$.*

3 Extended Dependency Spaces

This paper discusses an *existential* subclass of PBESs where universal-quantifiers are not allowed [1]. This class properly includes disjunctive PBESs [14]. Existential PBESs can be represented in simpler forms as shown in the next proposition.

Proposition 5 *For every existential PBES $\mathcal{E}$, there exists an existential PBES $\mathcal{E}'$ satisfying $[\![\mathcal{E}]\!](X) = [\![\mathcal{E}']\!](X)$ for all $X \in \mathrm{bnd}(\mathcal{E})$, and where $\mathcal{E}'$ is of the following form:*

$$\begin{array}{rcl}
\sigma_1 X_1(\mathbf{d}{:}\mathbf{D}) & = & \bigvee_{1\le k\le m_1} \exists \mathbf{e}{:}\mathbf{D}\ \varphi_{1k}(\mathbf{d},\mathbf{e}) \wedge X_{a_{1k1}}(\overrightarrow{f_{1k1}(\mathbf{d},\mathbf{e})}) \wedge \cdots \wedge X_{a_{1kp_{1k}}}(\overrightarrow{f_{1kp_{1k}}(\mathbf{d},\mathbf{e})}) \\
& \vdots & \\
\sigma_n X_n(\mathbf{d}{:}\mathbf{D}) & = & \bigvee_{1\le k\le m_n} \exists \mathbf{e}{:}\mathbf{D}\ \varphi_{nk}(\mathbf{d},\mathbf{e}) \wedge X_{a_{nk1}}(\overrightarrow{f_{nk1}(\mathbf{d},\mathbf{e})}) \wedge \cdots \wedge X_{a_{nkp_{nk}}}(\overrightarrow{f_{nkp_{nk}}(\mathbf{d},\mathbf{e})})
\end{array}$$

where σ_i is either μ or ν, $\overrightarrow{f_{ikj}(\mathbf{d},\mathbf{e})}$ is a sequence of data expressions possibly containing variables $\mathbf{d},\mathbf{e}$, and $\varphi_{ik}(\mathbf{d},\mathbf{e})$ is a Boolean expression containing no free variables except for $\mathbf{d},\mathbf{e}$.

In contrast, a disjunctive PBES is of the following form:

$$\begin{array}{rcl}
\sigma_1 X_1(\mathbf{d}{:}\mathbf{D}) & = & \bigvee_{1\le k\le m_1} \exists \mathbf{e}{:}\mathbf{D}\ \varphi_{1k}(\mathbf{d},\mathbf{e}) \wedge X_{a_{1k1}}(\overrightarrow{f_{1k1}(\mathbf{d},\mathbf{e})}) \\
& \vdots & \\
\sigma_n X_n(\mathbf{d}{:}\mathbf{D}) & = & \bigvee_{1\le k\le m_n} \exists \mathbf{e}{:}\mathbf{D}\ \varphi_{nk}(\mathbf{d},\mathbf{e}) \wedge X_{a_{nk1}}(\overrightarrow{f_{nk1}(\mathbf{d},\mathbf{e})})
\end{array}$$

We can easily see that disjunctive PBESs are subclass of existential PBESs. As a terminology, we use *k-th clause for X_i* to refer to $\exists \mathbf{e}{:}\mathbf{D}\ \varphi_{ik}(\mathbf{d},\mathbf{e}) \wedge X_{a_{ik1}}(\overrightarrow{f_{ik1}(\mathbf{d},\mathbf{e})}) \wedge \cdots \wedge X_{a_{ikp_{ik}}}(\overrightarrow{f_{ikp_{ik}}(\mathbf{d},\mathbf{e})})$.

Hereafter, we extend the notion of dependency spaces [14], which is designed for disjunctive PBESs, to those for existential PBESs. The dependency space for a PBES contains all its minimal proof graphs and hence is valuable to find a proof graph. The dependency space for a disjunctive PBES is a graph consisting of the vertices labelled with $X(\mathbf{v})$ for each data $\mathbf{v} \in \mathbb{D}^*$ and the edges $X_i(\mathbf{v}) \to X_j(\mathbf{w})$ for all dependencies meaning that $X_j(\mathbf{w}) \implies X_i(\mathbf{v})$. Here $X_j(\mathbf{w}) \implies X_i(\mathbf{v})$ means that the predicate formula $\varphi_i(\mathbf{v})$ of X_i holds under the assumption that $X_j(\mathbf{w})$ holds. A proof graph, if it exists, is found as its subgraph by seeking an infinite path satisfying a condition (2) of Definition 2. This corresponds to choosing one out-going edge for each vertex. In this sense, the dependency space consists of $\vee$-vertices. This framework makes sense because a disjunctive PBES contains exactly one predicate variable in each clause.

On the other hand an existential PBES generally contains more than one predicate variable in each clause $\exists \mathbf{e}{:}\mathbf{D}\ \varphi(\mathbf{d},\mathbf{e}) \wedge X_{a_1}(\overrightarrow{f_1(\mathbf{d},\mathbf{e})}) \wedge \cdots \wedge X_{a_p}(\overrightarrow{f_p(\mathbf{d},\mathbf{e})})$ defining X_i, which induces dependencies $X_{a_1}(\mathbf{w_1}) \wedge \cdots \wedge X_{a_p}(\mathbf{w_p}) \implies X_i(\mathbf{v})$ for any data $\mathbf{v},\mathbf{u} \in \mathbb{D}^*$ such that $\varphi(\mathbf{v},\mathbf{u})$ and $\mathbf{w_1} = \overrightarrow{f_1(\mathbf{v},\mathbf{u})}, \ldots, \mathbf{w_p} = \overrightarrow{f_p(\mathbf{v},\mathbf{u})}$. Hence $\wedge$-vertices are necessary. Therefore, we extend the notion of dependency spaces by introducing $\wedge$-vertices. Such dependencies vary according to the clauses. Thus, we need additional parameters i,k for $\wedge$-vertices in keeping track of the k-th clause of X_i. For these reasons, each $\wedge$-vertex is designed to be a quadruple $(i,k,\mathbf{v},\mathbf{u})$.

[1]This restriction can be relaxed so that universal-quantifiers emerge in φ_{ik} of Proposition 5, which does not affect the arguments of this paper.

We illustrate the idea by an example. Consider an existential PBES $\mathcal{E}_3$:

$$\nu X_1(n:N) \quad = \quad \exists n'{:}N\ \mathrm{even}(n) \land X_1(3n+5n') \land X_1(4n+5n')$$

The dependencies induced from the equation are $X_1(3n+5n') \land X_1(4n+5n') \implies X_1(n)$ for each $n' \in \mathbb{N}$ and even $n \in \mathbb{N}$. Observing the case $n=2$, the dependency $X_1(6+5n') \land X_1(8+5n') \implies X_1(2)$ exists for each $n' \in \mathbb{N}$. In order to show that $X_1(2)$ holds, it is enough that we choose one of these dependencies for constructing a proof graph. Suppose that we will show $X_1(2)$, we must find some n' such that both $X_1(6+5n')$ and $X_1(8+5n')$ hold. Thus it is natural to introduce a $\lor$-vertex $X_1(2)$ having edges to $\land$-vertices corresponding to n' values. Each $\land$-vertex has out-going edges to $X_1(6+5n')$ and $X_1(8+5n')$. This is represented in Figure 1, where $\lor$-vertices are oval and newly-introduced $\land$-vertices are rectangular. Each $\land$-vertex is labelled with $(i,k,\mathbf{v},\mathbf{w})$ where i and k come from k-th clause for X_i.

$$\begin{tikzcd}
 & & X_1(2) \arrow[dl] \arrow[d] \arrow[dr] & \\
 & \boxed{(1,1,2,0)} \arrow[dl] \arrow[d] & \boxed{(1,1,2,1)} \arrow[d] \arrow[dr] & \cdots \\
X_1(6) & X_1(8) & X_1(11) & X_1(13) \\
\vdots & \vdots & \vdots & \vdots
\end{tikzcd}$$

Figure 1: The dependency space of $\mathcal{E}_3$

Generally the extended graph consists of $\lor$-vertices $X_i(\mathbf{v})$ for all $1 \le i \le n$ and $\mathbf{v} \in \mathbb{D}^*$ and $\land$-vertices $(i,k,\mathbf{v},\mathbf{w})$ for all $1 \le i \le n$, k, $\mathbf{v} \in \mathbb{D}^*$, and $\mathbf{w} \in \mathbb{D}^*$. Each k-th clause $\exists \mathbf{e}{:}\mathbf{D}\ \varphi(\mathbf{d},\mathbf{e}) \land X_{a_1}(\overrightarrow{f_1(\mathbf{d},\mathbf{e})}) \land \cdots \land X_{a_p}(\overrightarrow{f_p(\mathbf{d},\mathbf{e})})$ for X_i constructs edges:

$$X_i(\mathbf{v}) \to (i,k,\mathbf{v},\mathbf{w}),\ \ (i,k,\mathbf{v},\mathbf{w}) \to X_{a_1}(\overrightarrow{f_1(\mathbf{v},\mathbf{w})}),\ \ \ldots,\ \ (i,k,\mathbf{v},\mathbf{w}) \to X_{a_p}(\overrightarrow{f_p(\mathbf{v},\mathbf{w})})$$

for every $\mathbf{v} \in \mathbb{D}^*$ and $\mathbf{w} \in \mathbb{D}^*$ such that $\varphi(\mathbf{v},\mathbf{w})$ holds (see the figure below).

$$\begin{tikzcd}
 & X_i(\mathbf{v}) \arrow[d] & \\
 & \boxed{(i,k,\mathbf{v},\mathbf{w})} \arrow[dl] \arrow[dr] \arrow[d, phantom, "\cdots"] & \\
X_{a_1}(\overrightarrow{f_1(\mathbf{v},\mathbf{w})}) & \cdots\cdots & X_{a_p}(\overrightarrow{f_p(\mathbf{v},\mathbf{w})})
\end{tikzcd}$$

From now on, we write dependency spaces to refer to the extended one by abbreviating "extended".

We formalize dependency spaces. The dependency space for a given PBES $\mathcal{E}$ is a labeled directed graph $G = (V, E, \Pi)$ such that

- $V = \mathrm{sig}(\mathcal{E}) \cup \mathbb{N} \times \mathbb{N} \times \mathbb{D}^* \times \mathbb{D}^*$ is a set of vertices,
- E is a set of edges with $E \subseteq V \times V$, which is determined from the above discussion, and
- $\Pi : V \to \{\lor, \land\}$ is a function which assigns $\lor/\land$ to all vertices.

For the dependency of the PBES $\mathcal{E}_3$, the next graph is a proof graph of $X_1(2)$. We get this proof graph by choosing $n' = 0$ for every vertex.

$$\begin{tikzcd}
X_1(2) \arrow[r] \arrow[dr] & X_1(6) \arrow[r] \arrow[dr] & X_1(18) \arrow[r] \arrow[dr] & \dots \\
 & X_1(8) \arrow[r] \arrow[dr] & X_1(24) \arrow[r] \arrow[dr] & \dots \\
 & & X_1(32) \arrow[r] & \dots
\end{tikzcd}$$

We can see that this proof graph is obtained by removing some vertices and collapsing $\wedge$-vertices into the vertex $X_1(v)$ from the dependency space of $\mathcal{E}_3$ (Figure 1).

We show that this property holds in general.

Lemma 6 *For a given existential PBES, if there exist proof graphs of $X(\mathbf{v})$, then one of them is obtained from its dependency space by removing some $\vee/\wedge$-vertices and collapsing $\wedge$-vertices $(i,k,\mathbf{v},\mathbf{w})$ into the vertex $X_i(\mathbf{v})$.*

Thus in order to obtain a proof graph from the dependency space, we encounter the problem that chooses one out-going edge for each $\vee$-node so that the condition (2) of Definition 2 is satisfied. This problem corresponds to a problem known as parity games (see Lemma 16 in the appendix for details, and parity games with finite nodes are decidable in NP. Moreover, there is a solver, named PGSolver [7], which efficiently solves many practical problems.

Unfortunately, since dependency spaces have infinite vertices, it is difficult to apply parity game solvers. Thus we need a way to reduce a dependency space to a finite one as shown in the next section.

4 Reduced Dependency Space

In this section, we extend reduced dependency spaces [15] to those for existential PBESs. We assume that an existential PBES $\mathcal{E}$ has the form of Proposition 5.

Given a PBES $\mathcal{E}$, we define functions $F_{ik} : \mathrm{sig}(\mathcal{E}) \to 2^B$ and $G_{ik} : B \to 2^{\mathrm{sig}(\mathcal{E})}$ for each k-th clause for X_i as follows, where B refers to $\mathbb{N}\times\mathbb{N}\times\mathbb{D}^*\times\mathbb{D}^*$:

$$F_{ik}(X_j(\mathbf{v})) = \begin{cases} \{(i,k,\mathbf{v},\mathbf{w}) \mid \varphi_{ik}(\mathbf{v},\mathbf{w})\} & \text{if } i = j \\ \emptyset & \text{otherwise} \end{cases}$$

$$G_{ik}(j,k',\mathbf{v},\mathbf{w}) = \begin{cases} \{X_{a_{ik1}}(\overrightarrow{f_{ik1}(\mathbf{v},\mathbf{w})}),\dots,X_{a_{ikp_{ik}}}(\overrightarrow{f_{ikp_{ik}}(\mathbf{v},\mathbf{w})})\} & \text{if } i = j \wedge k = k' \\ \emptyset & \text{otherwise} \end{cases}$$

Intuitively, F_{ik} is a function that takes a $\vee$-vertex $X_j(\mathbf{v})$ and returns $\wedge$-vertices as the successors. On the other hand, G_{ik} is a function that takes a $\wedge$-vertex $(j,k',\mathbf{v},\mathbf{w})$ and returns $\vee$-vertices as the successors. In other words, F_{ik} and G_{ik} indicate the dependencies.

A reduced dependency space is a graph divided by the congruence relation on the algebra that contains operators F_{ik}, G_{ik}. We formalize this relation.

Definition 7 *Let $\sim_D, \sim_B$ be an equivalence relation on* $\mathrm{sig}(\mathcal{E})$ *and B respectively. The pair of relations $\langle \sim_D, \sim_B \rangle$ is* feasible *if all these conditions hold:*

- *For all $i, j \in \mathbb{N}$, if $i \neq j$ then $X_i(\mathbf{v}) \nsim_D X_j(\mathbf{v}')$ for any $\mathbf{v}, \mathbf{v}' \in \mathbb{D}^*$.*
- *For all $i \in \mathbb{N}$ and $\mathbf{v}, \mathbf{v}' \in \mathbb{D}^*$, if $X_i(\mathbf{v}) \sim_D X_i(\mathbf{v}')$ then $F_{ik}(X_i(\mathbf{v})) \sim_B F_{ik}(X_i(\mathbf{v}'))$ for any k.*

- *For all $i, j \in \mathbb{N}$, if $i \neq j$ or $k \neq k'$ then $(i,k,\mathbf{v},\mathbf{w}) \not\sim_B (j,k',\mathbf{v}',\mathbf{w}')$ for any $\mathbf{v},\mathbf{v}',\mathbf{w},\mathbf{w}' \in \mathbb{D}^*$.*
- *For all $(i,k,\mathbf{v},\mathbf{w}),(i,k,\mathbf{v}',\mathbf{w}') \in B$, if $(i,k,\mathbf{v},\mathbf{w}) \sim_B (i,k,\mathbf{v}',\mathbf{w}')$ then $G_{ik}(i,k,\mathbf{v},\mathbf{w}) \sim_D G_{ik}(i,k,\mathbf{v}',\mathbf{w}')$.*

Here, we extend the notion of an equivalence relation $\sim$ on some set A for the equivalence relation on 2^A in this way:

$$\alpha,\beta \subseteq A.\ \alpha \sim \beta \text{ iff } \{[a]_\sim \mid a \in \alpha\} = \{[b]_\sim \mid b \in \beta\}$$

We define a reduced dependency space using a feasible pair of relations and dependency space.

Definition 8 *Let $G = (V,E,\Pi)$ be a dependency space of $\mathcal{E}$. For a feasible pair $\langle \sim_D, \sim_B \rangle$ of relations, an equivalence relation $\sim_G$ on V is defined as $\sim_D \cup \sim_B$. The* reduced dependency space *for a given feasible pair of relations is $G' = (V/{\sim_G}, E', \Pi/{\sim_G})$, where $E' = \{([v]_{\sim_G},[w]_{\sim_G}) \mid (v,w) \in E\}$.*

Note that $\Pi/{\sim_G}$ is well-defined from the definition of $\sim_G$.

Next theorem states that the membership problem is reduced to the problem finding a finite reduced dependency space.

Theorem 9 *Given a finite reduced dependency space of a PBES, then the membership problem of the PBES is decidable.*

5 Construction of Reduced Dependency Spaces

In this section, we propose a procedure to construct a feasible pair of relations, i.e., reduced dependency spaces, whose basic idea follows the one in [15]. This seems to be similar to minimization algorithm of automata, but the main difference is on that vertices are infinitely many. Here we use a logical formula to represent (possibly) infinitely many vertices in a single vertex. We start from the most degenerated vertices, which corresponds to a pair $\langle \sim_D, \sim_B \rangle$ of coarse equivalence relations, and divide each vertex until the pair becomes feasible. More specifically, we start from the $\vee$-vertices $\{X_1(\mathbf{v}) \mid \mathbf{v} \in \mathbb{D}^*\}, \ldots, \{X_n(\mathbf{v}) \mid \mathbf{v} \in \mathbb{D}^*\}$ and $\wedge$-vertices $\{(1,1,\mathbf{v},\mathbf{w}) \mid \mathbf{v},\mathbf{w} \in \mathbb{D}^*\}, \ldots, \{(n,m_n,\mathbf{v},\mathbf{w}) \mid \mathbf{v},\mathbf{w} \in \mathbb{D}^*\}$. The procedure keeps track of a partition of the set $\mathbb{D}^*$ for each $i \in \{1,\ldots,n\}$ and a partition of the set $\mathbb{D}^* \times \mathbb{D}^*$ for each $i \in \{1,\ldots,n\}, k \in \{1,\ldots,m_i\}$, and makes partitions finer.

Recall that a *partition* of a set A is a family Φ of sets satisfying $\bigcup_{\phi \in \Phi} \phi = A$ and $\forall \phi,\phi' \in \Phi.\ \phi \neq \phi' \implies \phi \cap \phi' = \emptyset$. For a given PBES $\mathcal{E}$, we call a family $\mathcal{P}$ of partitions is a *partition family of $\mathcal{E}$* if $\mathcal{P}$ has the form $\langle \Phi_1, \ldots, \Phi_n, \Psi_{11}, \ldots, \Psi_{nm_n} \rangle$, and satisfies the following conditions:

- Φ_i is a partition of $\mathbb{D}^*$ for every i, and
- Ψ_{ik} is a partition of $\mathbb{D}^* \times \mathbb{D}^*$ for every i,k.

Every element of a partition family $\mathcal{P}$ is a partition of $\mathbb{D}^*$ or $\mathbb{D}^* \times \mathbb{D}^*$, hence we naturally define an equivalence relation $\sim^D_{\mathcal{P}}$ on $\mathbb{D}^*$ and $\sim^B_{\mathcal{P}}$ on B.

We define a function H that takes a partition family and returns another partition family obtained by doing necessary division operations to its elements. The procedure repeatedly applies H to the initial partition family until it saturates. If it halts, the resulting tuple induces a reduced dependency space. In the procedure, functions $\mathbb{D}^* \to \mathbb{B}$ (resp. $\mathbb{D}^* \times \mathbb{D}^* \to \mathbb{B}$) represented by Boolean expressions with lambda binding are used to represent an infinite subset of a data domain $\mathbb{D}^*$ (resp. $\mathbb{D}^* \times \mathbb{D}^*$), In other words, a function $f = \lambda \mathbf{d}{:}D^*.\phi$ (resp. $g = \lambda(\mathbf{d},\mathbf{e}){:}D^* \times D^*.\phi$) can be regarded as a set $\{\mathbf{v} \in \mathbb{D}^* \mid f(\mathbf{v}) = \mathbb{t}\}$ (resp. $\{(\mathbf{v},\mathbf{w}) \in \mathbb{D}^* \times \mathbb{D}^* \mid g(\mathbf{v},\mathbf{w}) = \mathbb{t}\}$). In the sequel, we write Boolean functions for the corresponding sets.

The division function H consists of two steps, the division of Φ and Ψ. We give an intuitive explanation of the division $\Phi = \{\mathbb{N}\}$ by a set $f = \lambda d{:}N.\ \exists e{:}N\ d+e < 10$, where assuming that $d+e < 10$ appears in a PBES as $\varphi_{ik}(d,e)$. Recall that the function F_{ik} is defined by $F_{ik}(X_i(v)) = \{(i,k,v,w) \mid \varphi_{ik}(v,w)\}$ and the parameter e is quantified by $\exists$ in existential PBESs. We have to divide the data domain $\mathbb{N}$ into its intersection with f and the rest, i.e., $\mathbb{N} \cap f = \{v \in \mathbb{N} \mid v < 10\}$ and $\mathbb{N} \cap \overline{f} = \{v \in \mathbb{N} \mid v \geq 10\}$, where $\overline{f}$ denotes the complement of a set f. This division is necessary because the feasibility condition requires that the mapped values of F_{ik} are also in the same set, and hence we must separate $v \in \mathbb{N}$ according to whether $F_{ik}(X_i(v))$ is empty or not.

Next, suppose Φ is divided into two blocks $\{v \mid v < 10\}$ and $\{v \mid v \geq 10\}$ in the first step. We assume that a formula $X(d+e)$ appears in the predicate formula of a PBES. Then, we have to divide $\Psi = \{\mathbb{N} \times \mathbb{N}\}$ into $\{(v,w) \mid v+w < 10\}$ and $\{(v,w) \mid v+w \geq 10\}$. This is because the feasibility condition requires that the mapped values of G_{ik} are also in the same set.

Now we formalize these operations. We must divide each set ϕ in a partition Φ according to a set ψ in a partition Ψ, and similarly divide each set ψ in a partition Ψ according to a set ϕ in a partition Φ. For the definition, we prepare some kind of the inverse operation for F_{ik} and G_{ik}.

$$\begin{aligned} F'_{ik}(\psi) &= \{\mathbf{v} \mid (i,k,\mathbf{v},\mathbf{w}) \in F_{ik}(X_i(\mathbf{v})), (\mathbf{v},\mathbf{w}) \in \psi\} \\ G'_{ik}(\phi) &= \{(\mathbf{v},\mathbf{w}) \mid G_{ik}(i,k,\mathbf{v},\mathbf{w}) \cap \phi \neq \emptyset\} \end{aligned}$$

Then the division operations are given as follows:

$$\begin{aligned} \Phi \otimes^D_{ik} \psi &= \{F'_{ik}(\psi) \cap \phi \mid \phi \in \Phi\} \cup \{\overline{F'_{ik}(\psi)} \cap \phi \mid \phi \in \Phi\} \\ \Psi \otimes^B_{ik} \phi &= \{G'_{ik}(\phi) \cap \psi \mid \psi \in \Psi\} \cup \{\overline{G'_{ik}(\phi)} \cap \psi \mid \psi \in \Psi\} \end{aligned}$$

The operator $\otimes^D_{ik}$ obviously satisfies $(\Phi \otimes^D_{ik} \psi_1) \otimes^D_{ik} \psi_2 = (\Phi \otimes^D_{ik} \psi_2) \otimes^D_{ik} \psi_1$, thus we can naturally extend it on sets of formulas as follows:

$$\Phi \otimes^D_{ik} \{\psi_1, \ldots, \psi_p\} = \Phi \otimes^D_{ik} \psi_1 \otimes^D_{ik} \cdots \otimes^D_{ik} \psi_p$$

Also, it is easily shown that if Φ is a partition of $\mathbb{D}^*$, then $\Phi \otimes^D_{ik} \Psi$ is also a partition for a set Ψ' of formulas. These facts are the same in the case of operator $\otimes^B_{ik}$.

We unify these operators in a function that refines a given partition family.

Definition 10 *Let $\mathcal{P}$ be $\langle \Phi_1, \ldots, \Phi_n, \Psi_{11}, \ldots, \Psi_{nm_n} \rangle$. The partition functions H^D_{ik}, H^B_{ik} for each i and k are defined as follows:*

$$\begin{aligned} H^D_{ik}(\mathcal{P}) &= \langle \ldots, \Phi_{i-1}, \Phi_i \otimes^D_{ik} \Psi_{ik}, \Phi_{i+1}, \ldots \rangle \\ H^B_{ik}(\mathcal{P}) &= \langle \ldots, \Psi_{i(k-1)}, \Psi_{ik} \otimes^B_{ik} \Delta_D, \Psi_{i(k+1)}, \ldots \rangle \\ \Delta_D &= \bigcup_{1 \leq j \leq p_{ik}} \{\{X_{a_{ikj}}(\mathbf{v}) \mid \mathbf{v} \in \phi\} \mid \phi \in \Phi_{a_{ikj}}\} \end{aligned}$$

We bundle these functions as $H^D = H^D_{nm_n} \circ \cdots \circ H^D_{11}, H^B = H^B_{nm_n} \circ \cdots \circ H^B_{11}$ and $H = H^B \circ H^D$ by composition $\circ$.

We define the partition procedure that applies the partition function H to the trivial partition family $\mathcal{P}_0 = \langle \{\mathbb{D}^*\}, \ldots, \{\mathbb{D}^*\}, \{\mathbb{D}^* \times \mathbb{D}^*\}, \ldots, \{\mathbb{D}^* \times \mathbb{D}^*\} \rangle$ until it saturates. We write the family of the partitions obtained from the procedure as $H^\infty(\mathcal{P}_0)$.

Example 11 *Consider the existential PBES $\mathcal{E}_4$ given as follows:*

$$\nu X_1(n{:}N) \quad = \quad \exists n'{:}N \; \text{even}(n) \wedge X_1(3n+5n') \wedge X_1(4n+5n')$$

The reduced dependency space for $\mathcal{E}_4$ is:

$$\begin{tikzcd}
X_1(N_0) \arrow[d, shift left] \arrow[dr] & X_1(N_1) & & \\
\boxed{B_{00}} \arrow[u, shift left] & \boxed{B_{01}} \arrow[u] & \boxed{B_{10}} \arrow[ull] \arrow[ul] & \boxed{B_{11}} \arrow[ulll] \arrow[ull]
\end{tikzcd}$$

where $N_0 = \{0,2,\dots\}, N_1 = \{1,3,\dots\}$ and $B_{ik} = \{(1,1,d,e) \mid d \equiv_2 i \wedge e \equiv_2 k\}$ for each $i \in \{0,1\}$ and $k \in \{0,1\}$.

In order to construct this, we apply H to the initial partition $\langle\{\lambda n{:}N.true\},\{\lambda(n,n'){:}N^2.true\}\rangle$. First, we apply H^D:

$$\begin{array}{rcl} H^D(\langle\{\lambda n{:}N.true\},\{\lambda(n,n'){:}N^2.true\}\rangle) & = & (\langle\{\lambda n{:}N.true\}\otimes^D_{11}\{\lambda n{:}N.\mathrm{even}(n)\},\{\lambda(n,n'){:}N^2.true\}\rangle) \\ & = & \langle\{\lambda n{:}N.\mathrm{even}(n),\lambda n{:}N.\neg\mathrm{even}(n)\},\{\lambda(n,n'){:}N^2.true\}\rangle \end{array}$$

We write $\{\lambda n{:}N.\mathrm{even}(n),\lambda n{:}N.\neg\mathrm{even}(n)\}$ as Φ for readability. Next, we apply H^B:

$$\begin{array}{rl} H^B(\langle\Phi,\{\lambda(n,n'){:}N^2.true\}\rangle) = & \langle\Phi,\ \{\lambda(n,n'){:}N^2.true\}\otimes^B_{11}\Phi\rangle \\ = & \langle\Phi,\ \{\lambda(n,n'){:}N^2.\mathrm{even}(n)\wedge\mathrm{even}(n'),\lambda(n,n'){:}N^2.\mathrm{even}(n)\wedge\neg\mathrm{even}(n'), \\ & \quad \lambda(n,n'){:}N^2.\neg\mathrm{even}(n)\wedge\mathrm{even}(n'),\lambda(n,n'){:}N^2.\neg\mathrm{even}(n)\wedge\neg\mathrm{even}(n')\}\rangle \end{array}$$

The resulting partition family is a fixed-point of H, and hence the procedure stops. This partition family induces the set of vertices in the reduced dependency space.

We show that the procedure returns a partition family which induces a feasible pair of relations, i.e., reduced dependency space.

Theorem 12 *Suppose the procedure terminates and returns a partitions family $\mathcal{P} = H^\infty(\mathcal{P}_0)$. Then, the pair $\sim^D_{\mathcal{P}}$ and $\sim^B_{\mathcal{P}}$ is feasible.*

6 Implementation and an Example: Downsized McCarthy 91 Function

This section states implementation issues of the procedure presented in Section 5 and a bit more complex example, which is inspired by the McCarthy 91 function.

We describe the overview of our implementation. For a given PBES, the first step calculates a partitions family $\mathcal{P} = H^\infty(\mathcal{P}_0)$ by repeatedly applying the function H in Definition 10. This step requires a lot of SMT-solver calls. We'll explain the implementation in the next paragraph. Once the procedure terminates, the obtained partitions family $\mathcal{P}$ determines a feasible pair $\sim^D_{\mathcal{P}}$ and $\sim^B_{\mathcal{P}}$ by Theorem 12. Considering the finite dependency space constructed from the pair as a parity game, the second step constructs a proof graph by using PGSolver, which is justified by the proof of Theorem 9. (See Lemma 16 and Lemma 18 stating that the existence of a proof graph can be checked by solving a parity game on a reduced dependency space.)

The key to the implementation of $H^\infty(\mathcal{P}_0)$ is the operators $\otimes^D_{ik}$ and $\otimes^B_{ik}$ used in the function H. We focus on this and discuss how to implement these operators. We use Boolean expressions with lambda binding to represent subsets of data domains $\mathbb{D}^*$ and $\mathbb{D}^*\times\mathbb{D}^*$ as used for the intuitive explanation of the procedure and Example 11. Then it seems as if it would be simple to implement the procedure. It, however, induces non-termination without help of SMT solvers. Let us look more closely at the division operation $\Phi\otimes^D_{ik}\psi$. Suppose that $\lambda(\mathbf{d},\mathbf{e}){:}D^*\times D^*.\hat{\psi}$ is given as an argument of F'_{ik}. The resulting function is presented as $\lambda\mathbf{d}{:}D^*.\hat{\eta}$ where $\hat{\eta} = \exists\mathbf{e}{:}D^*(\varphi_{ik}(\mathbf{d},\mathbf{e})\wedge\hat{\psi})$. By using a set of functions to represent a partition Φ, each division of $\lambda\mathbf{d}{:}D^*.\hat{\phi}$ in Φ by $F'_{ik}(\psi)$ is simply implemented; it produces two functions $\lambda\mathbf{d}{:}D^*.(\hat{\eta}\wedge\hat{\phi})$ and $\lambda\mathbf{d}{:}D^*.(\neg\hat{\eta}\wedge\hat{\phi})$. This simple symbolic treatment always causes non-termination of the procedure without removing an empty set from the partition. Since the set represented by a function $\lambda\mathbf{d}{:}D^*.\hat{\phi}$ is empty if and only if $\hat{\phi}$ is unsatisfiable, this can be done by using an SMT solver. An incomplete unsatisfiability check easily causes a non-termination of the procedure, even if the procedure with complete unsatisfiability check terminates. Thus, the unsatisfiability check of Boolean expressions is one of the most important issues in implementing the procedure. For instance, the examples illustrated in this paper are all in the class of Presburger arithmetic, which is the first-order theory of the natural numbers with addition. It is known that the unsatisfiability check of Boolean expressions in this class is decidable [18].

Consider the following function F on $\mathbb{N}$ determined by a given $a \in \mathbb{N}$:

$$F(n) = \begin{cases} n-1 & \text{if } n > a \\ F(F(n+2)) & \text{if } n \leq a \end{cases}$$

The function $F(n)$ returns $n-1$ if $n > a$, and returns a otherwise. The latter property $F(n) = a$ for $n \leq a$ can be proved by induction on $n-k$ where $k \in \mathbb{N}$.

For an instance $a = 3$, this function can be modeled by the following existential PBES:

$$\begin{array}{rcl} \mu M(x{:}N, y{:}N) & = & (x > 3 \wedge y+1 = x \wedge X_T) \vee (\exists e{:}N\ x \leq 3 \wedge M(x+2, e) \wedge M(e, y)) \\ \nu X_T & = & X_T \end{array}$$

where X_T is a trivial predicate variable which denotes true.

To understand this modeling, we consider the case $x = 0$. In this case, because the first clause does not hold for any y, $M(0,y)$ holds only if $M(2,e)$ and $M(e,y)$ hold for some e. This implies $y = F(0)$ iff $\exists e{:}N\ e = F(0+2) \wedge y = F(e)$. In addition, in the case $x > 3$, $M(x,y)$ holds if $y+1 = x$, that is equivalent to $y = F(x)$. From this consideration, $M(x,y)$ holds if and only if $y = F(x)$.

To solve this example, we have implemented the procedure, which uses SMT solver Z3 [5] for deciding the emptiness of sets in the division. We attempted to solve the above PBES, and got the reduced dependency space consisting of 65 nodes in a few seconds. A proof graph is immediately found from the resulting space by applying PGSolver [7]. Figure 2 displays a part of the obtained graph consisting of vertices where $M(x,y)$ holds. Although a proof graph induced from the reduced dependency space is

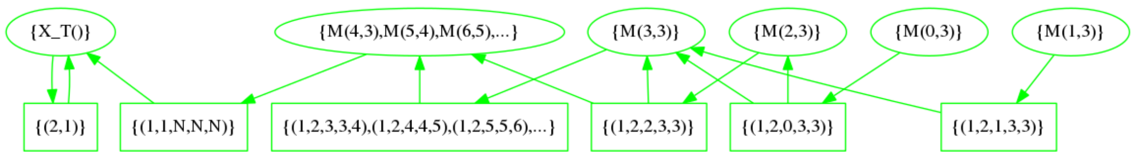

Figure 2: A part of the reduced dependency space where $M(x,y)$ holds.

finite, the reduced dependency space is nevertheless useful because the search space is infinite. We also tried to solve a larger instance $a = 10$ and got the spaces consisting of 394 nodes in 332 seconds, in which Z3 solver spent 325 seconds.

For another example, a disjunctive PBESs for a trading problem [15] is successfully solved by our implementation in a second, which produced a space consisting 12 nodes. Note that disjunctive PBESs is a subclass of existential PBESs.

In contrast, the procedure does not halt for the PBES $\mathcal{E}_2$ in Example 1, nor even for the next simple PBES:

$$\mu X(d{:}N) = (d = 0) \vee (d > 0 \wedge X(d-1))$$

where its solution is $X = \mathbb{N}$. Our procedure starts from the entire set $\mathbb{N}$ and divides it into $\{0\}$ and $\{1,2,\dots\}$ because of the first clause. After that, the latter set is split into $\{1\}$ and $\{2,3,\dots\}$ using the second clause. Endlessly, the procedure splits the latter set into the minimum number and the others. From this observation, the feasibility condition on $\langle \sim_D, \sim_B \rangle$ may be too strong, and weaker one is promising.

7 Conclusion

We have extended reduced dependency spaces for existential PBESs, and have shown that a proof graph is obtained from the space by solving parity games if the space is finite. Reduced dependency spaces are valuable because the dependency spaces for most of PBESs are infinite, but existential PBESs may have a finite reduced dependency space. We also have shown a procedure to construct reduced dependency spaces and have shown the correctness. We have shown some examples including a downsized McCarthy 91 function is successfully characterized by our method by applying an implementation.

Reduced dependency space is defined so that it contains all minimal proof graphs. For a membership problem to obtain a proof graph which proves $X(\mathbf{v})$, a reduced space may require too many division to make the entire data domain consistent. This sometimes induces the loss of termination of the procedure. Proposing a more clever procedure is one of the future works. Moreover, our implementation relies on the shape of PBESs, not sets defined by them. This is indicated by the example in the last of section 6. To clarify these conditions is also our future works.

References

[1] T. Chen, B. Ploeger, J. van de Pol & T. A. C. Willemse (2007): *Equivalence Checking for Infinite Systems Using Parameterized Boolean Equation Systems*. In: *Proceedings of the 18th International Conference on Concurrency Theory (CONCUR'07)*, *LNCS* 4703, Springer, Berlin, Heidelberg, pp. 120–135, doi:10.1007/978-3-540-74407-8_9.

[2] E. Clarke, O. Grumberg, S. Jha, Y. Lu & H. Veith (2003): *Counterexample-guided Abstraction Refinement for Symbolic Model Checking*. *Journal of ACM* 50(5), pp. 752–794, doi:10.1145/876638.876643.

[3] E. M. Clarke, Jr., O. Grumberg & D. A. Peled (1999): *Model Checking*. MIT Press, Cambridge, MA, USA.

[4] S. Cranen, B. Luttik & T. A. C. Willemse (2013): *Proof Graphs for Parameterised Boolean Equation Systems*. In: *Proc. of the 24th International Conference on Concurrency Theory, (CONCUR 2013)*, *LNCS* 8052, Springer, pp. 470–484, doi:10.1007/978-3-642-40184-8_33.

[5] L. De Moura & N. Bjørner (2008): *Z3: An Efficient SMT Solver*. In: *Proc. of the 14th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'08)*, *LNCS* 4963, Springer, Berlin, Heidelberg, pp. 337–340, doi:10.1007/978-3-540-78800-3_24.

[6] E. A. Emerson & C. S. Jutla (1991): *Tree automata, mu-calculus and determinacy*. In: *Foundations of Computer Science, 1991. Proceedings, 32nd Annual Symposium on*, IEEE, pp. 368–377, doi:10.1109/SFCS.1991.185392.

[7] O. Friedmann & M. Lange (2017): *The PGSolver Collection of Parity Game Solvers*. Available at `https://github.com/tcsprojects/pgsolver/blob/master/doc/pgsolver.pdf`.

[8] E. Grädel, P. G. Kolaitis, L. Libkin, M. Marx, J. Spencer, M. Y. Vardi, Y. Venema & S. Weinstein (2007): *Finite Model Theory and Its Applications*. Texts in Theoretical Computer Science. An EATCS Series, Springer, doi:10.1007/3-540-68804-8.

[9] S. Graf & H. Saidi (1997): *Construction of Abstract State Graphs with PVS*. In: *Proc. of the 9th International Conference Computer Aided Verification (CAV'97)*, *LNCS* 1254, Springer, pp. 72–83, doi:10.1007/3-540-63166-6_10.

[10] J. F. Groote & T. Willemse (2004): *Parameterised Boolean Equation Systems*. In: *Proc. of 15th International Conference on Concurrency Theory (CONCUR 2004)*, *LNCS* 3170, Springer, pp. 308–324, doi:10.1007/978-3-540-28644-8_20.

[11] J. F. Groote & T. A. C. Willemse (2005): *Model-checking Processes with Data*. *Science of Computer Programming* 56(3), pp. 251–273, doi:10.1016/j.scico.2004.08.002.

[12] J. F. Groote & T. A. C. Willemse (2005): *Parameterised Boolean Equation Systems*. Theoretical Computer Science 343(3), pp. 332–369, doi:10.1016/j.tcs.2005.06.016.

[13] J. F. Groote & T. A. C. Willemse (2004): *A Checker for Modal Formulae for Processes with Data*. In: Proc. of the 2nd International Symposium on Formal Methods for Components and Objects (FMCO 2003), LNCS 3188, Springer, pp. 223–239, doi:10.1007/978-3-540-30101-1_10.

[14] R. P. J. Koolen, T. A. C. Willemse & H. Zantema (2015): *Using SMT for Solving Fragments of Parameterised Boolean Equation Systems*. In: Proc. of the 13th International Symposium on Automated Technology for Verification and Analysis (ATVA 2015), LNCS 9364, Springer, pp. 14–30, doi:10.1007/978-3-319-24953-7_3.

[15] Y. Nagae, M. Sakai & H. Seki (2017): *An Extension of Proof Graphs for Disjunctive Parameterised Boolean Equation Systems*. Electronic Proceedings in Theoretical Computer Science 235, pp. 46–61, doi:10.4204/EPTCS.235.4.

[16] S. Orzan, W. Wesselink & T. A. C. Willemse (2009): *Static Analysis Techniques for Parameterised Boolean Equation Systems*. In: Proc. of the 15th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2009), LNCS 5505, Springer, pp. 230–245, doi:10.1007/978-3-642-00768-2_22.

[17] B. Ploeger, J. W. Wesselink & T. A. C. Willemse (2011): *Verification of Reactive Systems via Instantiation of Parameterised Boolean Equation Systems*. Information and Computation 209(4), pp. 637–663, doi:10.1016/j.ic.2010.11.025.

[18] M. Presburger (1929): *Über die Vollständigkeit eines gewissen Systems der Arithmetik ganzer Zahlen*. In: welchem die Addition als einzige Operation hervortritt, C. R. ler congrès des Mathématiciens des pays slaves, Warszawa, pp. 92–101.

A Proof of Proposition 5

Definition 13 Existential PBESs *are subclass of PBES defined by the following grammar:*

$$\begin{array}{rcl}\mathcal{E} & ::= & \emptyset \mid (\nu X(d:D) = \varphi)\mathcal{E} \mid (\mu X(d:D) = \varphi)\mathcal{E} \\ \varphi & ::= & b \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \exists d{:}D\ \varphi \mid X(\mathbf{exp})\end{array}$$

The difference from the original definition is that the universal-quantifier does not occur in φ.

Proposition 5 *For every existential PBES $\mathcal{E}$, there exists an existential PBES $\mathcal{E}'$ satisfying $[\![\mathcal{E}]\!](X) = [\![\mathcal{E}']\!](X)$ for all $X \in \mathrm{bnd}(\mathcal{E})$, and where $\mathcal{E}'$ is of the following form:*

$$\begin{array}{rcl}\sigma_1 X_1(\mathbf{d}{:}\mathbf{D}) & = & \bigvee_{1\leq k\leq m_1} \exists \mathbf{e}{:}\mathbf{D}\ \varphi_{1k}(\mathbf{d},\mathbf{e}) \wedge X_{a_{1k1}}(\overrightarrow{f_{1k1}(\mathbf{d},\mathbf{e})}) \wedge \cdots \wedge X_{a_{1kp_{1k}}}(\overrightarrow{f_{1kp_{1k}}(\mathbf{d},\mathbf{e})}) \\ & \vdots & \\ \sigma_n X_n(\mathbf{d}{:}\mathbf{D}) & = & \bigvee_{1\leq k\leq m_n} \exists \mathbf{e}{:}\mathbf{D}\ \varphi_{nk}(\mathbf{d},\mathbf{e}) \wedge X_{a_{nk1}}(\overrightarrow{f_{nk1}(\mathbf{d},\mathbf{e})}) \wedge \cdots \wedge X_{a_{nkp_{nk}}}(\overrightarrow{f_{nkp_{nk}}(\mathbf{d},\mathbf{e})})\end{array}$$

where σ_i is either μ or ν, $\overrightarrow{f_{ikj}(\mathbf{d},\mathbf{e})}$ is a sequence of data expressions possibly containing variables $\mathbf{d},\mathbf{e}$, and $\varphi_{ik}(\mathbf{d},\mathbf{e})$ is a Boolean expression containing no free variables except for $\mathbf{d},\mathbf{e}$.

Proof We can normalize each φ by the following steps.

(1) Rename all bound variables so that different scope variables are distinct.

(2) Lift up all existential quantifiers to obtain the form $\exists \mathbf{e}{:}D.\ \varphi'$.

(3) Calculate the disjunctive normal form of φ' and obtain $\bigvee_R \varphi_R$. Then $\bigvee_R \exists \mathbf{e}{:}D.\ \varphi_R$ is an expected form.

□

B Proofs related to proof graphs

B.1 Parity games and dependency spaces

Definition 14 *A* parity game *is a directed graph $G = (V, \rightarrow, \Omega, \Pi)$ such that*

- *V is a set of vertices,*
- *$\rightarrow$ is a set of edges with $\rightarrow \subseteq V \times V$,*
- *$\Omega : V \rightarrow \mathbb{N}$ is a function which assigns* priority *to each vertex, and*
- *$\Pi : V \rightarrow \{\circ, \Box\}$ is a function which assigns* player *to each vertex.*

Parity game is a game played by two players $\circ$ and $\Box$. This game progresses by moving the piece placed at a vertex along an edge. The player $\Pi(v)$ moves the piece when the piece is on a vertex v. We call a parity game started from v if the piece is initially placed on the vertex v. A play of a parity game is a sequence of vertices that the piece goes through. The player $\circ$ wins *the game if and only if the player $\Box$ cannot move the piece, or the largest priority that occurs infinitely often in the play is an even number when the play continues infinitely. We write* $\mathrm{rank}^{\infty}(\pi)$ *as this largest priority for the play π.*

For a given parity game G started from a fixed vertex, a strategy S of a player is a function that selects a vertex to move along an edge, given a vertex owned by the player. A strategy S is called a *winning strategy* when the player wins the game G according to S even if the opponent player plays optimally.

The next proposition shows the determinacy of the winner for parity games.

Proposition 15 ([6]) *Given a parity game, starting vertex dominates the winner, which means that either of the player $\circ$ or $\square$ has a winning strategy according to the vertex started from.*

We say that *a player wins the game on a vertex* if he has a winning strategy started from the vertex.

A dependency space for a PBES $\mathcal{E}$ is regarded as a parity game. The function Π naturally defined from the shape of $\vee/\wedge$-vertices, i.e., $\Pi(v) = \circ$ if and only if v is an $\vee$-vertex. We give the priority function Ω as follows:

$$\Omega(v) = \begin{cases} u - \mathrm{rank}(X_i), & \text{if } v = X_i(\mathbf{v}) \in \mathrm{sig}(\mathcal{E}) \\ 0, & \text{otherwise} \end{cases}$$

where u is the minimum even number satisfying $u \geq \mathrm{rank}(X_i)$ for any $X_i \in \mathrm{bnd}(\mathcal{E})$. We call this *a parity game obtained from $\mathcal{E}$*. Note that V and $\rightarrow$ in dependency spaces are generally infinite sets, while $\{\Omega(v) \mid v \in V\}$ is finite. Thus each parity game corresponding to a dependency space is well-defined.

Moreover, a reduced dependency space obtained from a feasible relation $\sim$ is also regarded as a parity game, because $\Omega/\!\sim$ and $\Pi/\!\sim$ are well-defined. We call this *a reduced parity game $G/\!\sim$* for a parity game G and a feasible relation $\sim$.

B.2 Proof of Lemma 6

This lemma is justified by Lemma 16.

Lemma 16 *For a given PBES $\mathcal{E}$, there exists a proof graph that proves $X(\mathbf{v}_0)$ if and only if the player $\circ$ wins the game G, obtained from $\mathcal{E}$, on $X(\mathbf{v}_0)$.*

Proof We use the notation $X_i(\mathbf{v})^\bullet$ to represent the postset of $X_i(\mathbf{v})$ for the target graph.

$\Rightarrow$) Let P be a proof graph of $X(\mathbf{v}_0)$. By the form of existential PBESs and the condition (1) of proof graphs, for every vertex $X_i(\mathbf{v})$ in P there exist k and $\mathbf{w}$ such that

$$\varphi_{ik}(\mathbf{v},\mathbf{w}) \text{ and } X_i(\mathbf{v})^\bullet \supseteq \{X_{a_{ik1}}(f_{ik1}(\mathbf{v},\mathbf{w})),\ldots,X_{a_{ikp_{ik}}}(f_{ikp_{ik}}(\mathbf{v},\mathbf{w}))\}.$$

From this, we take the strategy of the player $\circ$: when the piece is on a vertex $X_i(\mathbf{v}) \in P$, move the piece to $(i,k,\mathbf{v},\mathbf{w})$ for $k,\mathbf{w}$ determined by the above condition.

We show that the piece never goes on the $\vee$-vertices not in P with the parity game started from $X(\mathbf{v_0})$ by induction of the steps of the game. First, the piece is on $X(\mathbf{v}_0)$, and it is in P. Assume the piece is on $X_i(\mathbf{v}) \in P$. Let $(i,k,\mathbf{v},\mathbf{w})$ be the $\wedge$-vertex which the piece goes on under the strategy. From the definition of the strategy, all vertices to which the player $\square$ can move the piece are in P. Thus, the piece never goes on the $\vee$-vertices not in P, and the strategy is well-defined.

We show that the strategy is winning strategy started from $X(\mathbf{v}_0)$. If the piece cannot move on a vertex $X_i(\mathbf{v})$, then $\varphi_{ik}(\mathbf{v},\mathbf{w})$ never holds for any $k,\mathbf{w}$ by the definition of G. Then, it contradicts the condition (1) of proof graphs. In addition, suppose that there exists an infinite parity game in which the player $\circ$ loses. Let π be the infinite path tracing the movement of the piece. We define an infinite path π' from π by collapsing the $\wedge$-vertices into $\vee$-vertices. Because π' only consists of $\vee$-vertices, π' is also a path of P. P is a proof graph, therefore, the minimum rank of X^∞ with π' is even by the condition (2) of proof graphs. In contrast, the largest priority on π' is odd because the player $\circ$ loses. The minimum rank with π' is odd by the definition of the priority function Ω, however, this contradicts.

$\Leftarrow$) Assume that the player $\circ$ has a winning strategy. We construct a graph P:

- $X(\mathbf{v}_0)$ is in P.
- Suppose that $X_i(\mathbf{v})$ is in P and $(i,k,\mathbf{v},\mathbf{w})$ is the vertex to which the player $\circ$ moves the piece on $X_i(\mathbf{v})$. Then, $(i,k,\mathbf{v},\mathbf{w})^\bullet$ is in P and there exist edges from $X_i(\mathbf{v})$ to each vertex in $(i,k,\mathbf{v},\mathbf{w})^\bullet$.

It is obvious that the graph P have the conditions of proof graphs. $\square$

B.3 Proof of Theorem 9

Proposition 17 *Let G be a parity game and $G/\!\sim$ be a reduced parity game for a PBES. Then, for all vertex $[A]_\sim, [B]_\sim \in G/\!\sim$ and for all $u \in [A]_\sim$, $[A]_\sim \to [B]_\sim$ implies $\exists v \in [B]_\sim.\ u \to v$.*

The theorem 9 holds immediate from the following lemma and the fact that solving a finite parity game is decidable.

Lemma 18 *The player $\circ$ wins on $X(\mathbf{v})$ for a dependency space if and only if the player $\circ$ wins on $[X(\mathbf{v})]_\sim$ for a reduced dependency space.*

Proof $\Rightarrow$) We prove contraposition. Suppose the player $\circ$ loses on $[X(\mathbf{v})]_\sim$. This means that the player $\square$ wins on $[X(\mathbf{v})]_\sim$ by the proposition 15. Let S' be a winning strategy of the player $\square$ on $[X(\mathbf{v})]_\sim$. Then, a strategy S of the player $\square$ on $X(\mathbf{v})$ can be defined as $S(u) = v$ for $u \in G$ where $v \in S'([u]_\sim)$ from the proposition 17.

Any game P on $X(\mathbf{v})$ according to the strategy S is corresponding to a game P' on $[X(\mathbf{v})]_\sim$ according to the strategy S'. That is, the sequence of the rank P is equal to the sequence of P'. S' is a winning strategy, therefore S is also a winning strategy.

$\Leftarrow$) Suppose the player $\circ$ has a winning strategy S' on $[X(\mathbf{v})]_\sim$. We can define a strategy S on $X(\mathbf{v})$ using S' in a similar way, and S is also a winning strategy. $\square$

B.4 Proof of Theorem 12

Theorem 12 *Suppose the procedure terminates and returns a partitions family $\mathcal{P} = H^\infty(\mathcal{P}_0)$. Then, the pair $\sim^D_\mathcal{P}$ and $\sim^B_\mathcal{P}$ is feasible.*

Proof Proof by contradiction. Let $\mathcal{P} = \langle \ldots, \Phi^\infty_i, \ldots, \Psi^\infty_{ik}, \ldots \rangle$ be a fixed-point of H and $\sim_\mathcal{P} = \sim^D_\mathcal{P} \cup \sim^B_\mathcal{P}$. If $\mathcal{P}$ is not feasible, then at least one of the following conditions holds:

(1) There exists $X_i(\mathbf{v})$ and $X_i(\mathbf{v}')$ such that $X_i(\mathbf{v}) \sim_\mathcal{P} X_i(\mathbf{v}')$ and $F_{ik}(X_i(\mathbf{v})) \not\sim_\mathcal{P} F_{ik}(X_i(\mathbf{v}'))$ for some k.

(2) There exists $(i,k,\mathbf{v},\mathbf{w})$ and $(i,k,\mathbf{v}',\mathbf{w}')$ such that $(i,k,\mathbf{v},\mathbf{w}) \sim_\mathcal{P} (i,k,\mathbf{v}',\mathbf{w}')$ and $G_{ik}(i,k,\mathbf{v},\mathbf{w}) \not\sim_\mathcal{P} G_{ik}(i,k,\mathbf{v}',\mathbf{w}')$.

Suppose the condition (1) holds. Then, w.l.o.g., $\bigcup_{x \in F_{ik}(X_i(\mathbf{v}'))}[x]_{\sim_\mathcal{P}} \cap [(i,k,\mathbf{v},\mathbf{w})]_{\sim_\mathcal{P}} = \emptyset$ for some $(i,k,\mathbf{v},\mathbf{w}) \in F_{ik}(X_i(\mathbf{v}))$ by the definition of feasible relation. In particular, because $\bigcup_{x \in F_{ik}(X_i(\mathbf{v}'))}[x]_{\sim_\mathcal{P}} \supseteq F_{ik}(X_i(\mathbf{v}'))$, $F_{ik}(X_i(\mathbf{v}')) \cap [(i,k,\mathbf{v},\mathbf{w})]_{\sim_\mathcal{P}} = \emptyset$. Let $\psi = [(i,k,\mathbf{v},\mathbf{w})]_{\sim_\mathcal{P}} \in \Psi^\infty_{ik}$. We have $F_{ik}(X_i(\mathbf{v})) \cap \psi \supseteq \{(i,k,\mathbf{v},\mathbf{w})\} \neq \emptyset$ and $F_{ik}(X_i(\mathbf{v}')) \cap \psi = \emptyset$. Therefore, $X_i(\mathbf{v})$ and $X_i(\mathbf{v}')$ belong different block when $\otimes^D_{ik}$ splits $\mathcal{P}$. This contradicts that $\mathcal{P}$ is a fixed-point of H.

Moreover, suppose the condition (2) holds. By a similar argument, there exists $X_a(d_a) \in G_{ik}(i,k,\mathbf{v},\mathbf{w})$ such that $G_{ik}(i,k,\mathbf{v}',\mathbf{w}') \cap [X_a(d_a)]_{\sim_\mathcal{P}} = \emptyset$. Recall $G_{ik}(i,k,\mathbf{v},\mathbf{w}) = \{X_{a_1}(f_1(\mathbf{v},\mathbf{w})), \ldots, X_{a_p}(f_p(\mathbf{v},\mathbf{w}))\}$, $a = a_q$ and $d_a = f_q(\mathbf{v},\mathbf{w})$ for some $1 \leq q \leq p$. Let $\phi = [X_a(d_a)]_{\sim_\mathcal{P}} \in \Phi_a$. Obviously, $G_{ik}(i,k,\mathbf{v},\mathbf{w}) \cap \phi \neq \emptyset$ and $G_{ik}(i,k,\mathbf{v}',\mathbf{w}') \cap \phi = \emptyset$. Thus, $(i,k,\mathbf{v},\mathbf{w})$ and $(i,k,\mathbf{v}',\mathbf{w}')$ belong different block when $\otimes^B_{ik}$ splits $\mathcal{P}$. This contradicts $\mathcal{P}$ is a fixed-point. $\square$