Reduced dependency spaces for existential parameterised Boolean equation systems* (Extended Abstract)†

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. This system is regarded as a declarative program defining functions that take a datum and returns a Boolean value. The membership problem of PBESs is a problem to decide whether a given element is in the defined set or not, which corresponds to an execution of the program. It is known that the problem, is undecidable in general, is reduced to the existence of proof graphs. This paper proposes a subclass of PBESs which expresses $\forall$-quantifiers free formulas, and studies a technique to solve the problem on it.

To check the existence of a proof graph, we introduce a dependency space which is a graph containing all of the 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 no finite dependency space except for reduced one. 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 a downsized McCarthy 91 function.

1 Introduction

A *Parameterised Boolean Equation System* (PBES) [12, 9, 11] 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 [12, 10], 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 as fixed-points that satisfy the equations, where μ (resp. ν) represents the least (resp. greatest) fixed-point. In the PBES $\mathcal{E}_1$, Y is an empty set since Y is the least fixed-point 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 fixed-point satisfying that $X(n)$ iff $X(n+1) \vee Y(n)$ for any $n \geq 0$.

The membership problem for PBESs is undecidable in general, since it is shown how a μ-calculus formula and a process algebraic specification, both involving data parameters, can be transformed into a PBES [12]. Some techniques have been proposed to solve the problem for some subclasses of PBESs:

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

†A long version will be found at [14].

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 that possibly produces a finite reduced proof graph, which represents an infinite proof graph [15]. The technique manages the subclass of 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 space designed for it. We clarify some relevance between extended dependency spaces and the existence of proof graphs. Dependency spaces are, however, infinite graphs in most cases. Thus we reduce dependency spaces finite for the existential class in a more sophisticated way based on the existing technique. 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 [11] 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 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 of 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 contains no free predicate variables as well as no 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:*

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

Since the definition of the semantics is complex, we will explain it by an example, where the formal definition is found in [11]. 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 $\{1,3,5,\ldots\}$ 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 and simplify the equation, which results in "$\nu X_1(n:N) = (n = 0 \wedge X_1(n+2)) \vee (n > 0 \wedge \mathrm{odd}(n-1) \wedge X_1(0) \wedge X_1(n+2))$". 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 $X_1 = \{0,2,4,\ldots\}$ and $X_2 = \{1,3,5,\ldots\}$. 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. Specifically for 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 are 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 $\{v' \in V \mid v \to v'\}$.

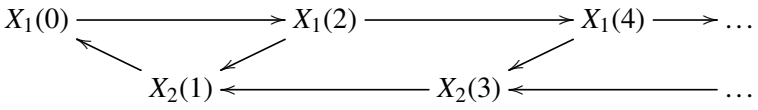
Definition 2 *Let $\mathcal{E}$ be a PBES, $V \subseteq \mathrm{sig}(\mathcal{E})$, $\to \subseteq V \times V$, and $r \in \mathbb{B}$. The tuple $\langle V, \to, r \rangle$ is called a* proof graph *for the PBES if both of the following conditions hold:*

(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:*



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 $\forall$-quantifiers are not allowed [1]. This class properly includes disjunctive PBESs [13]. 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 has the following form:*

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

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 $\mathbf{d},\mathbf{e}$.

We use a terminology *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 [13], which is designed for disjunctive PBESs, to those for existential PBESs. The dependency space for a PBES contains all its 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 $\wedge$-vertex. Such dependencies vary according to the clauses. Thus, we need additional parameters i,k for $\wedge$-vetices in keeping track of k-th clause of X_i. For these reasons, each $\wedge$-vertex is designed to be a quadruple $(i,k,\mathbf{v},\mathbf{u})$.

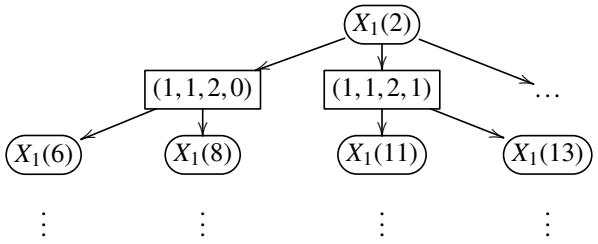
We illustrate the idea by an example. Consider an existential PBES $\mathcal{E}_3$:

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

The dependencies induced from the equation are $X_1(3n+5n') \wedge X_1(4n+5n') \implies X_1(n)$ for any $n' \in \mathbb{N}$ and any even $n \in \mathbb{N}$. Observing the case $n = 2$, the dependency $X_1(6+5n') \wedge X_1(8+5n') \implies X_1(2)$ exist for any $n' \in \mathbb{N}$. 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

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

$X_1(6+5n')$ and $X_1(8+5n')$ hold. Thus it is natural to introduce a $\vee$-vertex $X_1(2)$ having edges to $\wedge$-vertices corresponding to n' values. Each $\wedge$-vertex has out-going edges to $X_1(6+5n')$ and $X_1(8+5n')$. This is represented as follows:

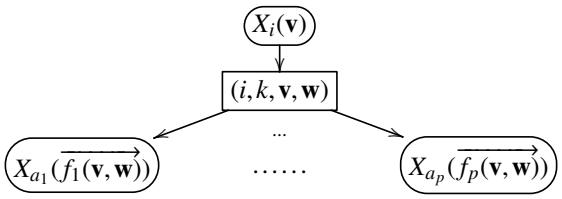


where $\vee$-vertices are oval and newly-introduced $\wedge$-vertices are rectangular. Each $\wedge$-vertex is labelled with $(i,k,\mathbf{v},\mathbf{w})$ where i and k come from k-th clause for X_i.

Generally the extended graph consists of $\vee$-vertices $X_i(\mathbf{v})$ for all $1 \leq i \leq n$ and $\mathbf{v} \in \mathbb{D}^*$ and $\wedge$-vertices $(i,k,\mathbf{v},\mathbf{w})$ for all $1 \leq i \leq n$, k, $\mathbf{v} \in \mathbb{D}^*$, and $\mathbf{w} \in \mathbb{D}^*$. Each k-th clause $\exists \mathbf{e}{:}\mathbf{D}\ \varphi(\mathbf{d},\mathbf{e}) \wedge X_{a_1}(\overrightarrow{f_1(\mathbf{d},\mathbf{e})}) \wedge \cdots \wedge X_{a_p}(\overrightarrow{f_p(\mathbf{d},\mathbf{e})})$ 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 $\mathbf{v} \in \mathbb{D}$ and $\mathbf{w} \in \mathbb{D}$ such that $\varphi(\mathbf{v},\mathbf{w})$ holds (see the figure below).



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
- $\Pi : V \to \{\vee, \wedge\}$ is a function which assigns $\vee/\wedge$ to every vertices.

It is shown that any proof graph for an existential PBES 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 satisfies. This problem corresponds to a problem known as parity games. Moreover, parity games with finite nodes have a good algorithm [6].

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 finite 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 $\mathbb{N} \times \mathbb{N} \times \mathbb{D}^* \times \mathbb{D}^*$:

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

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

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

Intuitively, 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 6 *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 $X_i(\mathbf{v}) \not\sim_D X_j(\mathbf{v}')$ 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 extends 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 7 *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 8 *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 is similar to minimization algorithm of an automata. 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 keep track 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 make partitions finer.

Remember 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 similary divide each set ψ in a partition Ψ according to a set ϕ in a partition Φ. For the definition, we prepare kind of the inverse operation for F_{ik} and G_{ik}.

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

Then the division operations are given as follows:

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

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 operator $\otimes^B_{ik}$.

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

Definition 9 *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:*

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

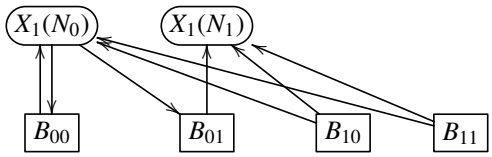
We bundle these functions as $H^D = H^D_{nm_n} \circ \cdots \circ H^D_{11}, H^B = H^B_{nm_n} \circ \cdots \circ H^B_{11}$ *and* $H = H^B \circ H^D$ *by composition* $\circ$.

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

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

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



where $N_0 = \{0,2,\ldots\}, N_1 = \{1,3,\ldots\}$ *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*:*

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

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 11 *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 McCarthy 91 function.

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 10. 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 see more closely focusing on 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, 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 $a \in \mathbb{N}$:

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

It is not trivial but $F(n)$ returns $n-1$ if given n greater than a, and returns a otherwise.

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

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

where X_T is a redundant 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 [6]. The figure 1 displays a part of the obtained graph consisting of vertices where player $\circ$ wins, i.e., $M(x, y)$ holds. Although a proof graph induced from

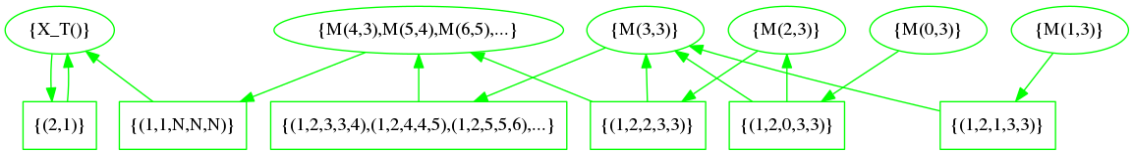


Figure 1: A part of the reduced dependency space where $M(x, y)$ holds.

the reduced dependency space is 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 trivial 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 divide 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 premising.

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 that 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 proof graph. For membership problem to obtain a proof graph which proves $X(\mathbf{v})$, a reduced space may require too much division to make 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.

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

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] Jan Friso Groote & Tim 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.

[13] 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.

[14] Y. Nagae & M. Sakai (2017): *Reduced Dependency Spaces for Existential Parameterised Boolean Equation Systems.* Available at `https://www.trs.css.i.nagoya-u.ac.jp/~sakai/papers/nagae-sakai-wpte2017.pdf`. A long version of this paper.

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

[16] S. Orzan, W. Wesselink & T. AC 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.