Computing Parameterized Invariants of Parameterized Petri Nets

Javier Esparza, Mikhail Raskin and Christoph Welzel

Technical University Munich

Abstract. A fundamental advantage of Petri net models is the possibility to automatically compute useful system invariants from the syntax of the net. Classical techniques used for this are place invariants, P-components, siphons or traps. Recently, Bozga et al. have presented a novel technique for the *parameterized* verification of safety properties of systems with a ring or array architecture. They show that the statement "for every instance of the parameterized Petri net, all markings satisfying the linear invariants associated to all the P-components, siphons and traps of the instance are safe" can be encoded in WS1S and checked using tools like MONA. However, while the technique certifies that this infinite set of linear invariants extracted from P-components, siphons or traps are strong enough to prove safety, it does not return an explanation of this fact understandable by humans. We present a CEGAR loop that constructs a *finite* set of *parameterized* P-components, siphons or traps, whose infinitely many instances are strong enough to prove safety. For this we design parameterization procedures for different architectures.

1 Introduction

A fundamental advantage of Petri net system models is the possibility to automatically extract useful system invariants from the syntax of the net at low computational cost. Classical techniques used for this purpose are place invariants, P-components, siphons or traps [41, 42, 20]. All of them are syntactic objects that can be computed using linear algebra or boolean logic, and from which semantic linear invariants can be extracted. For example, from the fact that a set of places Q is an initially marked trap of the net one extracts the linear invariant $\sum_{p \in Q} M(Q) \geq 1$, which is satisfied for every reachable marking M. This information can be used to prove safety properties: Given a set $\mathcal{S}$ of safe markings, if every marking satisfying the invariants extracted from a set of objects is safe, then all reachable markings are safe.

Classical net invariants have been very successfully used in the verification of single systems [10, 27, 13], or as complement to state-space exploration [46]. Recently, an extension of this idea to the *parameterized* verification of safety properties of systems with a ring or array architecture has been presented in [16, 15]. The parameterized verification problem asks whether a system composed of n processes is safe for every $n \geq 2$ [11, 25, 4]. Bozga *et al.* show in [16, 15] that the statement

"For every instance of the parameterized system, all markings satisfying the linear invariants associated to all the P-components, siphons and traps of the corresponding Petri net is safe"

can be encoded in Weak Second-order Logic With One Successor (WS1S), or its analogous WS2S for two successors. This means that the statement holds iff its formula encoding is valid. This problem is decidable, and highly optimized tools exist for it, like MONA [44, 37]. The method of [15] is not complete (i.e., there are safe systems for which the invariants derived from P-components, siphons and traps are not strong enough to prove safety), but it succeeds for a remarkable set of examples. Further, incompleteness is inherent to every algorithmic method, since safety of parameterized nets is undecidable even if processes only manipulate data from a bounded domain [5, 11].

While the technique of [16, 15] is able to prove interesting properties of numerous systems, it does not yet provide an explanation of why the property holds. Indeed, when the technique succeeds for a given parameterized Petri net, the user only knows that the set of all invariants deduced from siphons, traps, and P-components together are strong enough to prove safety. However, the technique does not return a minimal set of these invariants. Moreover, since the parameterized Petri net has infinitely many instances, such a set contains infinitely many invariants. In this paper we show how to overcome this obstacle. We present a technique that automatically computes a *finite* set of *parameterized invariants*, readable by humans. This is achieved by lifting a CEGAR (counterexample-guided abstraction refinement) loop, introduced in [28] and further developed in [27, 29, 12], to the parameterized case. Each iteration of the loop of [28, 27] first computes a counterexample, i.e., a marking that violates the desired safety property but satisfies all invariants computed so far, and then computes a P-component, siphon, or trap showing that the marking is not reachable. If no counterexample exists the property is established, and if no P-component, siphon or trap can be found the method fails. The technique is implemented on top of an SMT-solver, which receives as input a linear constraint describing the set of safe markings, and iteratively computes the set of linear invariants derived from P-components, siphons, and traps.

If we naively lift the CEGAR loop to the parameterized case, the loop never terminates. Indeed, since the loop computes one new invariant per iteration, and infinitely many invariants are needed to prove correctness of all instances, termination is not possible. So we need a procedure to extract from one single invariant for one instance a *parameterized* invariant, i.e., an infinite set of invariants for all instances, finitely represented as a WS1S-formula. We present a *semi-automatic* and an *automatic approach*. In the semi-automatic approach the user guesses the parameterized invariant, and automatically checks it, using the WS1S-checker. The automatic approach does not need user interaction, but only works for systems with symmetric structure. We provide automatic procedures for rings and barrier crowds, a class of systems closely related to broadcast protocols. We present experimental results on a number of systems. The semi-

automatic approach requires more human interaction, but produces smaller sets of invariants.

Related work. The parameterized verification problem has been extensively studied for systems whose associated transition systems are well-structured [36, 1, 33] (see e.g. [4] for a survey). In this case the verification problem reduces to a coverability problem, for which different algorithms exist [14, 35, 43, 32]; the marking equation (which is roughly equivalent to place invariants) have also been applied [6]. However, the transition systems of parametric rings and arrays are typically not well-structured.

Parameterized verification of ring and array systems has also been studied in a number of papers. Three popular techniques are regular model checking (see e.g. [40, 3, 2]), abstraction [8, 9], and automata learning [18]. All of them apply symbolic state-space exploration to try to compute a finite automaton recognizing the set of reachable markings of all instances, or an abstraction thereof. Our technique avoids any state-space exploration. Also, symbolic state-space exploration techniques are not geared towards providing explanations. Indeed, while the set of reachable markings of all instances is the strongest invariant of the system, it is also one single monolithic invariant, typically difficult to interpret by human users. Our CEGAR loop aims at finding a *collection* of invariants, each of them simple and interpretable.

Many works in parameterized follow the cut-off approach, where one manually proves a *cut-off* bound $c \geq 2$ such that correctness for at most c processes implies correctness for any number of processes (see e.g. [17, 23, 22, 7, 38], and [11] for a survey). It then suffices to prove the property for systems of up to c processes, which can be done using finite-state model checking techniques. Compared to this technique, ours is fully automatic.

Full version. Due to space restrictions we refer to the full version [30] for proofs and extended explanations.

2 Preliminaries

WS1S. Formulas of WS1S over first-order variables $\boldsymbol{x}, \boldsymbol{y}, \ldots$ and second-order variables $\boldsymbol{X}, \boldsymbol{Y}, \ldots$ have the following syntax:

$$t := \boldsymbol{x} \mid 0 \mid \mathrm{succ}(t) \qquad \text{(terms)}$$
$$\phi := t_1 \leq t_2 \mid \boldsymbol{x} \in \boldsymbol{X} \mid \phi_1 \wedge \phi_2 \mid \neg\phi_1 \mid \exists \boldsymbol{x} \colon \phi \mid \exists \boldsymbol{X} \colon \phi \qquad \text{(formulas)}$$

An interpretation assigns elements of $\mathbb{N}_0 = \{0, 1, 2, 3, \ldots\}$ to first order variables and *finite* subsets of $\mathbb{N}_0$ to second-order variables. Given an interpretation, the semantics that assigns numbers to terms and truth values to formulas is defined in the usual way.

We extend the syntax with constants $0, 1, 2, 3, \ldots$, and terms of the form $\boldsymbol{x} + c$ with $c \in \mathbb{N}_0$. Further, a term $\boldsymbol{x} \oplus_{\boldsymbol{n}} 1$ in a formula φ stands for

$$(\boldsymbol{x} + 1 < \boldsymbol{n} \wedge \varphi[\boldsymbol{x} \oplus_{\boldsymbol{n}} 1 \leftarrow \boldsymbol{x} + 1]) \vee (\boldsymbol{n} = \boldsymbol{x} + 1 \wedge \varphi[\boldsymbol{x} \oplus_{\boldsymbol{n}} 1 \leftarrow 0])$$

where $\varphi[t \leftarrow t']$ denotes the result of substituting t' for t in φ. The terms $\boldsymbol{x} \oplus_{\boldsymbol{n}} c$ for every $1 \leq c$ are defined similarly. We let $\varphi(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_\ell, \boldsymbol{X}_1, \ldots, \boldsymbol{X}_k)$ denote that φ uses at most $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_\ell$ and $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_k$ as free first-order resp. second-order variables. Finally, we also make liberal use of the following macros:

$$
\begin{array}{ll}
\boldsymbol{X} = \emptyset & \forall \boldsymbol{x} \colon \neg(\boldsymbol{x} \in \boldsymbol{X}) \\
\boldsymbol{X} = \{\boldsymbol{x}\} & \boldsymbol{x} \in \boldsymbol{X} \wedge \forall \boldsymbol{y} \colon \boldsymbol{y} \in \boldsymbol{X} \to \boldsymbol{y} = \boldsymbol{x} \\
\boldsymbol{X} = [\boldsymbol{n}] & \forall \boldsymbol{x} \colon \boldsymbol{x} \in \boldsymbol{X} \leftrightarrow \boldsymbol{x} < \boldsymbol{n} \\
\boldsymbol{X} \cap \boldsymbol{Y} = \emptyset & \forall \boldsymbol{x} \colon \neg(\boldsymbol{x} \in \boldsymbol{X} \vee \boldsymbol{x} \in \boldsymbol{Y}) \\
|\boldsymbol{X}| = 1 & \exists \boldsymbol{x} \colon \boldsymbol{X} = \{\boldsymbol{x}\} \\
|\boldsymbol{X}| \leq 1 & \boldsymbol{X} = \emptyset \vee |\boldsymbol{X}| = 1 \\
\boldsymbol{X} = \overline{\boldsymbol{Y}} & \forall \boldsymbol{x} \colon \boldsymbol{x} \in \boldsymbol{X} \leftrightarrow \neg(\boldsymbol{x} \in \boldsymbol{Y}) \\
\boldsymbol{Y} = \boldsymbol{X} \oplus_{\boldsymbol{n}} 1 & \forall \boldsymbol{x} \colon \boldsymbol{x} \oplus_{\boldsymbol{n}} 1 \in \boldsymbol{Y} \leftrightarrow \boldsymbol{x} \in \boldsymbol{X}
\end{array}
$$

with "stands for" between the two columns.

Petri Nets. We use a presentation of Petri nets equivalent to but slightly different from the standard one. A *net* is a pair $\langle P, T \rangle$ where P is a nonempty, finite set of *places* and $T \subseteq 2^P \times 2^P$ is a set of *transitions*. Given a transition $t = \langle P_1, P_2 \rangle$, we call P_1 the *preset* and *postset* of t, respectively. We also denote P_1 by ${}^\bullet t$ and P_2 by $t^\bullet$. Given a place p, we denote by ${}^\bullet p$ and $p^\bullet$ the sets of transitions $\langle P_1, P_2 \rangle$ such that $p \in P_2$ and $p \in P_1$, respectively. Given a set X of places or transitions, we let ${}^\bullet X := \bigcup_{x \in X} {}^\bullet x$ and $X^\bullet := \bigcup_{x \in X} x^\bullet$.

A *marking* of $N = \langle P, T \rangle$ is a function $M \colon P \to \mathbb{N}$. A Petri net is a pair $\langle N, M \rangle$, where N is a net and M is the *initial marking* of N. A transition $t = \langle P_1, P_2 \rangle$ is enabled at a marking M if $M(p) \geq 1$ for every $p \in P_1$. If t is enabled at M then it can *fire*, leading to the marking M' given by $M'(p) = M(p) + 1$ for every $p \in P_2 \setminus P_1$, $M'(p) = M(p) - 1$ for every $p \in P_1 \setminus P_2$, and $M'(p) = M(p)$ otherwise. We write $M \xrightarrow{t} M'$, and $M \xrightarrow{\sigma} M'$ for a finite sequence $\sigma = t_1 t_2 \ldots t_n$ if there are markings $M_1, \ldots, M_n$ such that $M \xrightarrow{t_1} M_1 \xrightarrow{t_2} \cdots M_{n-1} \xrightarrow{t_n} M'$. M' is reachable from M if $M \xrightarrow{\sigma} M'$ for some sequence σ.

A marking M is *1-bounded* if $M(p) \leq 1$ for every place p. A Petri net is *1-bounded* if every marking reachable from the initial marking is 1-bounded. A 1-bounded marking M of a Petri net is also defined by the set of marked places; i.e., $\wr M \wr = \{p \in P \colon M(p) = 1\}$.

3 Parameterized Petri Nets

Definition 1 (Parameterized nets). *A parameterized net is a pair $\mathcal{N} = \langle \mathcal{P}, Tr \rangle$, where $\mathcal{P}$ is a finite set of place names and $Tr(\boldsymbol{n}, \mathcal{X}, \mathcal{Y})$ is a WS1S-formula over one first-order variable $\boldsymbol{n}$ which represents the considered size of the instance and two tuples $\mathcal{X}$ and $\mathcal{Y}$ of second-order variables for each place name of $\mathcal{P}$; i.e., for a fixed enumeration $p_1, \ldots, p_k$ of the elements of $\mathcal{P}$ we get $\mathcal{X} = \langle \mathcal{X}_{p_i} \rangle_{i=1}^k$ and $\mathcal{Y} = \langle \mathcal{Y}_{p_i} \rangle_{i=1}^k$. We call such tuples of variables placeset variables.*

Let $[n] = \{0, \ldots, n-1\}$. A parameterized net $\mathcal{N}$ induces a net $\mathcal{N}(n) = \langle P_n, T_n \rangle$ for every $n \geq 1$, where $P_n = \mathcal{P} \times [n]$ (i.e., P_n consists of n copies of $\mathcal{P}$),

and T_n contains a transition $\langle P_1, P_2 \rangle$ for every pair $P_1, P_2 \subseteq P_n$ of sets of places such that "$Tr(n, P_1, P_2)$" holds. More formally, this means that $\mu \models Tr$ for an interpretation μ s.t. $\mu(\boldsymbol{n}) = n$, $\mu(\mathcal{X}_p) = \{i \in [n] : \langle p, i \rangle \in P_1\}$, and $\mu(\mathcal{Y}_p) = \{i \in [n] : \langle p, i \rangle \in P_2\}$ for all $p \in \mathcal{P}$. Therefore, the intended meaning of $Tr(n, \mathcal{X}, \mathcal{Y})$ is "the pair $\langle \mathcal{X}, \mathcal{Y} \rangle$ of placesets is (the preset and postset of) a transition if the size is n". We say that $\mathcal{N}(n)$ is an *instance* of $\mathcal{N}$.

In the following we use $\langle p, i \rangle$ and $p(i)$ as equivalent notations for the elements of $P_n = \mathcal{P} \times [n]$.

Example 1. We consider a version of the dining philosophers. Philosophers and forks are numbered 0, 1, ..., $n - 1$. For every $i > 0$ the i-th philosopher first grabs the i-th and then the $(i \oplus_n 1)$-th fork, where $\oplus_n$ denotes addition modulo n. Philosopher 0 proceeds the other way round: she first grabs fork 1, and then fork 0. After eating all philosophers return their forks in one single step. We formalize this in the following parameterized net $\mathcal{N} = \langle \mathcal{P}, Tr \rangle$:

- $\mathcal{P} = \{\text{think}, \text{wait}, \text{eat}, \text{free}, \text{taken}\}$. Intuitively, $\{\text{think}(i), \text{wait}(i), \text{eat}(i)\}$ are the states of the i-th philosopher, and $\{\text{free}(i), \text{taken}(i)\}$ the states of the i-th fork.
- $Tr(\boldsymbol{n}, \mathcal{X}, \mathcal{Y}) = \text{GrabFirst} \vee \text{GrabSecond} \vee \text{Release}$. We only present the formula for GrabFirst, the complete description can be found in the appendix.

$$
\text{GrabFirst} :=
\begin{pmatrix}
\exists \boldsymbol{x} . \ 1 \leq \boldsymbol{x} < \boldsymbol{n} \wedge (\mathcal{X}_{\text{think}} = \mathcal{X}_{\text{free}} = \mathcal{Y}_{\text{wait}} = \mathcal{Y}_{\text{taken}} = \{\boldsymbol{x}\}) \\
\wedge (\mathcal{X}_{\text{wait}} = \mathcal{X}_{\text{eat}} = \mathcal{X}_{\text{taken}} = \emptyset) \\
\wedge (\mathcal{Y}_{\text{think}} = \mathcal{Y}_{\text{eat}} = \mathcal{Y}_{\text{free}} = \emptyset)
\end{pmatrix}
$$
$$
\vee
$$
$$
\begin{pmatrix}
(\mathcal{X}_{\text{think}} = \mathcal{Y}_{\text{taken}} = \{0\}) \wedge (\mathcal{X}_{\text{free}} = \mathcal{Y}_{\text{wait}} = \{1\}) \\
\wedge (\mathcal{X}_{\text{wait}} = \mathcal{X}_{\text{eat}} = \mathcal{X}_{\text{taken}} = \emptyset) \\
\wedge (\mathcal{Y}_{\text{think}} = \mathcal{Y}_{\text{eat}} = \mathcal{Y}_{\text{free}} = \emptyset)
\end{pmatrix}
$$

Intuitively, the preset of GrabFirst is a philosopher in state think and her left (resp. right fork for philosopher 0) in state free; the postset puts the philosopher in state wait and the fork in state taken. The instance $\mathcal{N}(3)$ is shown in Figure 1.

Parameterized Petri nets are parameterized nets with a WS1S-formula defining its initial markings:

Definition 2 (Parameterized Petri nets). *A parameterized Petri net is a pair $\langle \mathcal{N}, Initial \rangle$, where $\mathcal{N}$ is a parameterized net, and $Initial(\boldsymbol{n}, \mathcal{M})$ is a WS1S-formula over a first-order variable $\boldsymbol{n}$ and a placeset variable $\mathcal{M}$.*

A parameterized Petri net defines an infinite family of Petri nets. Loosely speaking, a Petri net $\langle N, M \rangle$ belongs to the family if N is an instance of $\mathcal{N}$, i.e., $N = \mathcal{N}(n)$ for some $n \geq 1$, and M is a 1-bounded marking of N satisfying $Initial(n, \mathcal{M})$. For example, if $\mathcal{P} = \{p_1, p_2\}$, $n = 3$ and $Initial(\{0, 1\}, \{0, 2\})$ holds, then the family contains a Petri net $\langle \mathcal{N}(3), M_3 \rangle$ such that M_3 is a 1-bounded marking with $\lfloor M_3 \rceil = \{p_1(0), p_1(1), p_2(0), p_2(2)\}$.

Fig. 1. $\mathcal{N}(3)$ for Example 1. Places which are colored green are initially marked w.r.t. *Initial*($\mathcal{X}$) from Example 2. Note the repeating structure for philosophers 1 and 2 while philosopher 0 grabs her forks in the opposite order. We abbreviate think(i) to th(i), and similarly with the other states.
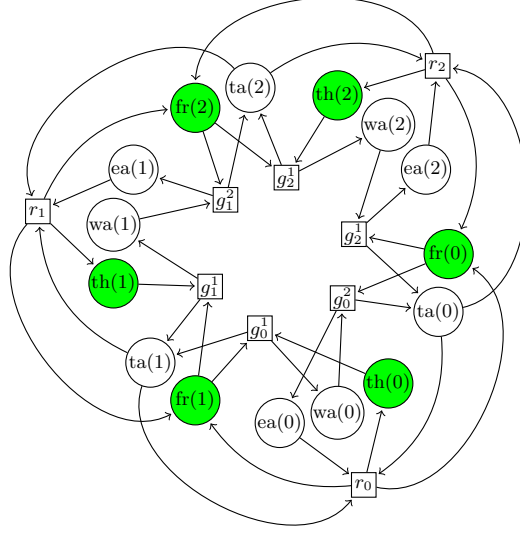
Example 2. The family of initial markings in which all philosophers think and all forks are free is modeled by:

$$Initial(\boldsymbol{n}, \mathcal{M}) := (\mathcal{M}_{\text{think}} = \mathcal{M}_{\text{free}} = [\boldsymbol{n}]) \wedge (\mathcal{M}_{\text{wait}} = \mathcal{M}_{\text{eat}} = \mathcal{M}_{\text{taken}} = \emptyset).$$

Example 3. Let us now model a simple version of the readers/writers system. A process can be idle, reading, or writing. An idle process can start to read if no other process is writing, and it can start to write if every other process is idle. We obtain the parameterized net $\mathcal{N} = \langle \mathcal{P}, Tr \rangle$, where

- $\mathcal{P} = \{\text{idle}, \text{rd}, \text{wr}, \text{not\_wr}\}$.
- $Tr(\boldsymbol{n}, \mathcal{X}, \mathcal{Y}) = \text{StartR} \vee \text{StopR} \vee \text{StartW} \vee \text{StopW}$. We give the formulae StartR and StartW, the other two being simpler.

$$\text{StartR} := \exists \boldsymbol{x} \ . \ \begin{pmatrix} (\mathcal{X}_{\text{idle}} = \{\boldsymbol{x}\} \wedge \mathcal{X}_{\text{not\_wr}} = \overline{\mathcal{X}_{\text{idle}}} \wedge (\mathcal{X}_{\text{rd}} = \mathcal{X}_{\text{wr}} = \emptyset) \\ \wedge \ \mathcal{Y}_{\text{rd}} = \{\boldsymbol{x}\} \wedge \mathcal{Y}_{\text{not\_wr}} = \overline{\mathcal{X}_{\text{idle}}} \wedge (\mathcal{Y}_{\text{idle}} = \mathcal{Y}_{\text{wr}} = \emptyset) \end{pmatrix}$$

$$\text{StartW} := \exists \boldsymbol{x} \ . \ \begin{pmatrix} \mathcal{X}_{\text{idle}} = [\boldsymbol{n}] \wedge \mathcal{X}_{\text{not\_wr}} = \{\boldsymbol{x}\} \wedge (\mathcal{X}_{\text{rd}} = \mathcal{X}_{\text{wr}} = \emptyset) \\ \wedge \ \mathcal{Y}_{\text{idle}} = [\boldsymbol{n}] \setminus \{\boldsymbol{x}\} \wedge \mathcal{Y}_{\text{wr}} = \{\boldsymbol{x}\} \wedge (\mathcal{Y}_{\text{rd}} = \mathcal{Y}_{\text{not\_wr}} = \emptyset) \end{pmatrix}$$

So the preset of a StartR transition is $\{\mathrm{idle}(i), \mathrm{not\_wr}(0), \ldots, \mathrm{not\_wr}(n-1)\}$ for some i, and the postset is $\{\mathrm{rd}(i), \mathrm{not\_wr}(0), \ldots, \mathrm{not\_wr}(n-1)\}$. The initial markings in which every process is initially idle are modeled by:

$$Initial(\boldsymbol{n}, \mathcal{X}) := \mathcal{X}_{\mathrm{idle}} = [\boldsymbol{n}] \wedge \mathcal{X}_{\mathrm{not\_wr}} = [\boldsymbol{n}] \wedge (\mathcal{X}_{\mathrm{rd}} = \mathcal{X}_{\mathrm{wr}} = \emptyset)$$

Observe that in the dining philosophers transitions have presets and postsets of size 3, independently of the number of philosophers. On the contrary, in the readers and writers problems the transitions of $\mathcal{N}(n)$ have presets and postsets of size n. Intuitively, our formalism allows to model transitions involving all processes or, for example, all even processes. Observe also that in both cases the formula $Initial$ has exactly one model for every $n \geq 1$, but this is not required.

Proving deadlock-freedom for the dining philosophers. Let us now give a taste of what our paper achieves for Example 1. It is well known that this version of the dining philosophers is deadlock-free. However, finding a proof based on parameterized invariants of the systems is not so easy. Using the semi-automatic use of the approach we present we find the following five invariants to do so. The fully automatic analysis of this example gives ten properties of the system which collectively induce deadlock-freedom.

The first two invariants simply express that at every reachable marking M, and for every $0 \leq i \leq n-1$, the i-th philosopher is either thinking, waiting, or eating, and the i-th fork is either free or taken.

$$M(\mathrm{think}(i)) + M(\mathrm{wait}(i)) + M(\mathrm{eat}(i)) = 1 \tag{1}$$

$$M(\mathrm{free}(i)) + M(\mathrm{taken}(i)) = 1. \tag{2}$$

The last three invariants provide the key insights. The last one holds for every $1 \leq i \leq n-2$:

$$M(\mathrm{wait}(0)) + M(\mathrm{eat}(0)) + M(\mathrm{free}(1)) + M(\mathrm{wait}(1)) + M(\mathrm{eat}(1)) = 1 \tag{3}$$

$$M(\mathrm{eat}(0)) + M(\mathrm{free}(0)) + M(\mathrm{eat}(n-1)) = 1 \tag{4}$$

$$M(\mathrm{eat}(i)) + M(\mathrm{eat}(i+1)) + M(\mathrm{free}(i+1)) + M(\mathrm{wait}(i+1)) = 1 \tag{5}$$

Let us sketch why (1)-(5) imply deadlock freedom. Let P_i denote the i-th philosopher and F_i the i-th fork. If P_0 is eating, then F_0 and F_1 are taken by (1)-(4), and there is no deadlock because P_0 can return them. The same holds if P_1 is eating by (1)-(3) and (5), or if any of $P_2, \ldots, P_{n-1}$ is eating by (1)-(2) and (5). If no philosopher eats, then by (1)-(3) and (5) either P_{i+1} is thinking and F_{i+1} is free for some $i \in \{1, \ldots, n-2\}$, or P_{i+1} is waiting for every $i \in \{1, \ldots, n-2\}$. In the first case P_{i+1} can grab F_{i+1}. In the second case P_{n-1} is waiting, and since F_0 is free by (1)-(2) and (4), it can grab F_0.

4 Checking 1-boundedness

Our techniques work for parameterized Petri nets whose instances are 1-bounded. We present a technique that automatically checks 1-boundedness of all our ex-

amples. We say that a set of places Q of a Petri net $\langle N, M \rangle$, where $N = \langle P, T \rangle$, is

- *1-balanced* if for every transition $\langle P_1, P_2 \rangle \in T$ either $|P_1 \cap Q| = 1 = |P_2 \cap Q|$, or $|P_1 \cap Q| = 0 = |P_2 \cap Q|$, or $|P_1 \cap Q| \geq 2$.
- *1-bounded* at M if $M(Q) \leq 1$.

The following proposition is an immediate consequence of the definition:

Proposition 1. *If Q is a 1-balanced and 1-bounded set of places of $\langle N, M \rangle$, then $M'(Q) = M(Q)$ holds for every reachable marking M'.*

We abbreviate "1-bounded and 1-balanced set" to 1BB-set, and say that N is *covered* by 1BB-sets if every place belongs to some 1BB-set at initial marking M. By the proposition above, if N is covered by 1BB-sets at M, then $M'(p) \leq 1$ holds for every reachable marking M' and every place p, and so N is 1-bounded.

Given a parameterized Petri net $(\mathcal{N}, \mathit{Initial})$, we can check if all instances are covered by 1BB-sets with the following formula:

$$\mathit{1Bal}(\boldsymbol{n}, \mathcal{X}) := \forall \mathcal{Y}, \mathcal{Z} \colon \mathit{Tr}(\boldsymbol{n}, \mathcal{Y}, \mathcal{Z}) \rightarrow \quad \begin{aligned} &(|\mathcal{X} \cap \mathcal{Y}| = 0 = |\mathcal{X} \cap \mathcal{Z}|) \vee \\ &(|\mathcal{X} \cap \mathcal{Y}| = 1 = |\mathcal{X} \cap \mathcal{Z}|) \vee \\ &(|\mathcal{X} \cap \mathcal{Y}| > 1) \end{aligned}$$

$$\mathit{1Bnd}(\boldsymbol{n}, \mathcal{X}, \mathcal{M}) := |\mathcal{X} \cap \mathcal{M}| \leq 1$$
$$\mathit{Cover} := \forall \boldsymbol{n}, \forall \mathcal{M} \colon \mathit{Initial}(\boldsymbol{n}, \mathcal{M}) \rightarrow (\forall \boldsymbol{x} \colon \exists \mathcal{X} \colon \boldsymbol{x} \in \mathcal{X} \wedge$$
$$\mathit{1Bal}(\boldsymbol{n}, \mathcal{X}) \wedge$$
$$\mathit{1Bnd}(\boldsymbol{n}, \mathcal{X}, \mathcal{M}))$$

Observe that if Q is a 1BB-set then at every reachable marking exactly one of the places of Q is marked, with exactly one token. The sets of places corresponding to a philosopher, a fork, a reader, or a writer are 1BB-sets. Unsurprisingly, all our parameterized Petri net models are covered by 1BB-sets. Moreover, this can be automatically proved in a few seconds by checking the formula above. This gives us an automatic proof that all the Petri nets we consider are 1-bounded.

5 Checking safety properties

Let $\langle \mathcal{N}, \mathit{Initial} \rangle$ be a parameterized Petri net, and let $\mathit{Safe}(\boldsymbol{n}, \mathcal{M})$ be a WS1S-formula describing a set of "safe" markings of the instances of $\mathcal{N}$ (for example, "safe" could mean deadlock-free). It is easy to prove (using simulations of Turing machines by Petri nets like those of [24]) that the existence of some unsafe reachable marking in some instance of a given parameterized Petri net $\langle \mathcal{N}, \mathit{Initial} \rangle$ is undecidable. In [15, 16] we describe a semi-algorithm for the problem that derives from $\langle \mathcal{N}, \mathit{Initial} \rangle$ a formula $\mathit{PReach}(\boldsymbol{n}, \mathcal{M})$ describing a superset of the set of reachable markings of all instances, and checks that the formula

$$\mathit{SafetyCheck} := \forall \boldsymbol{n} \forall \mathcal{M} \colon \mathit{PReach}(\boldsymbol{n}, \mathcal{M}) \rightarrow \mathit{Safe}(\boldsymbol{n}, \mathcal{M})$$

holds. We recall the main construction of [15, 16], adapted and expanded.

1BB-sets again. Recall that if a marking M' of some instance $\langle N, M \rangle$ of a net $\langle \mathcal{N}, Initial \rangle$ is reachable from M, then $M'(Q) \leq 1$ holds for every 1BB-set of places Q of $\langle N, M \rangle$. So this latter property can be interpreted as a test for potential reachability: Only markings that pass the test can be reachable. We introduce a formula $1BBTest(\boldsymbol{n}, \mathcal{M}', \mathcal{M})$ expressing that $\mathcal{M}'$ passes the test with respect to $\mathcal{M}$ (i.e., $\mathcal{M}'$ might be reachable from $\mathcal{M}$).

$$1BBTest(\boldsymbol{n}, \mathcal{M}', \mathcal{M}) := \forall \mathcal{X} \colon \begin{pmatrix} 1Bal(\boldsymbol{n}, \mathcal{X}) \\ \wedge 1Bnd(\boldsymbol{n}, \mathcal{X}, \mathcal{M}) \end{pmatrix} \to 1Bnd(\boldsymbol{n}, \mathcal{X}, \mathcal{M}')$$

Siphons and traps. Let $\langle N, M \rangle$ be a Petri net with $N = \langle P, T \rangle$ and let $Q \subseteq P$ be a set of places. Q is a *trap* of N if $^\bullet Q \subseteq Q^\bullet$, and a *siphon* of N if $Q^\bullet \subseteq {}^\bullet Q$.

- If Q is a siphon and $M(Q) = 0$, then $M'(Q) = 0$ for all markings M' reachable from M.
- If Q is a trap and $M(Q) \geq 1$, then $M'(Q) \geq 1$ for all markings M' reachable from M.

If M' is reachable from M then it satisfies the following property: $M'(Q) \geq 1$ for every trap Q such that $M(Q) \geq 1$. A marking satisfying this property *passes the trap test* for $\langle N, M \rangle$. We construct a formula $TrapTest(\boldsymbol{n}, \mathcal{M})$ expressing that $\mathcal{M}$ passes the trap test for some instance of a parameterized Petri net. We first introduce a formula expressing that a set $\mathcal{X}$ of places is a trap.

$$Trap(\boldsymbol{n}, \mathcal{X}) := \forall \mathcal{Y}, \mathcal{Z} \colon (Tr(\boldsymbol{n}, \mathcal{Y}, \mathcal{Z}) \wedge \mathcal{X} \cap \mathcal{Y} \neq \emptyset) \to \mathcal{X} \cap \mathcal{Z} \neq \emptyset$$

Now we have:

$$Marked(\boldsymbol{n}, \mathcal{X}, \mathcal{M}) := \mathcal{X} \cap \mathcal{M} \neq \emptyset$$
$$TrapTest(\boldsymbol{n}, \mathcal{M}', \mathcal{M}) := \forall \mathcal{X} \colon \begin{pmatrix} Trap(\boldsymbol{n}, \mathcal{X}) \\ \wedge Marked(\boldsymbol{n}, \mathcal{X}, \mathcal{M}) \end{pmatrix} \to Marked(\boldsymbol{n}, \mathcal{X}, \mathcal{M}')$$

Similarly we obtain a formula for a siphon test:

$$Empty(\boldsymbol{n}, \mathcal{X}, \mathcal{M}) := \mathcal{X} \cap \mathcal{M} = \emptyset$$
$$SiphTest(\boldsymbol{n}, \mathcal{M}', \mathcal{M}) := \forall \mathcal{X} \colon \begin{pmatrix} Siphon(\boldsymbol{n}, \mathcal{X}) \\ \wedge Empty(\boldsymbol{n}, \mathcal{X}, \mathcal{M}) \end{pmatrix} \to Empty(\boldsymbol{n}, \mathcal{X}, \mathcal{M}')$$

We can now give the formula *PReach*:

$$PReach(\boldsymbol{n}, \mathcal{M}', \mathcal{M}) := \begin{pmatrix} 1BBTest(\boldsymbol{n}, \mathcal{M}', \mathcal{M}) \\ \wedge TrapTest(\boldsymbol{n}, \mathcal{M}', \mathcal{M}) \\ \wedge SiphTest(\boldsymbol{n}, \mathcal{M}', \mathcal{M}) \end{pmatrix}$$
$$PReach(\boldsymbol{n}, \mathcal{M}') := \exists \mathcal{M} \colon Initial(\boldsymbol{n}, \mathcal{M}) \wedge PReach(\boldsymbol{n}, \mathcal{M}', \mathcal{M})$$

5.1 Automatic computation of parameterized invariants

In [15] it was shown that many safety properties of parameterized Petri nets can be proved to hold *for all instances* by checking validity of the corresponding *PReach* formula. However, the technique does not return a set of invariants strong enough to prove the property. In this section we show how to overcome this problem. We design a CEGAR loop which, when successful, yields a finite set of *parameterized* invariants that imply the safety property being considered.

We proceed as follows. In the first part of the section, we describe a CEGAR loop for the non-parameterized case. The input to the procedure is a parameterized Petri net $\langle \mathcal{N}, \mathit{Initial} \rangle$ and a number n such that all reachable markings of all instances $\mathcal{N}(1), \ldots, \mathcal{N}(n)$ are safe. The output is a set of invariants of $\mathcal{N}(1), \ldots, \mathcal{N}(n)$, derived from balanced sets, siphons, and traps, which are strong enough to prove safety. Since the set of all balanced placesets, siphons, and traps of these instances is finite, the procedure is guaranteed to terminate even if it computes one invariant at a time. Then we modify the loop by inserting an additional *parameterization procedure* that exploits the regularity of $\langle \mathcal{N}, \mathit{Initial} \rangle$. The procedure transforms a balanced placeset (siphon, trap) of a particular instance, say $\mathcal{N}(4)$, into a possibly infinite set of balanced placesets (siphons, traps) of all instances, encoded as the set of models of a WS1S-formula. This formula is a finite representation of the infinite set.

For the sake of brevity, in the rest of the section we describe a CEGAR loop that only constructs traps. This allows us to avoid numerous repetitions of the phrase "balanced placesets, siphons, and traps". Since the structure of the loop is completely generic, this is purely a presentation issue without loss of generality[1].

A CEGAR loop for the non-parameterized case. We need some preliminaries. Let $\mathcal{N} = \langle \mathcal{P}, \mathit{Tr} \rangle$ be a parameterized Petri net, and let $\mathcal{X}$ be a placeset variable. An *interpretation* of $\mathcal{X}$ is a pair $\mathbf{X} = \langle \ell, Q \rangle$, where $\ell \geq 1$ and Q is a set of places of $\mathcal{N}(\ell)$. We identify $\mathbf{X}$ and the tuple $\langle \mathbf{X}_p \rangle_{p \in \mathcal{P}}$, where $\mathbf{X}_p \subseteq [\ell]$, defined by $j \in \mathbf{X}_p$ iff $p(j) \in Q$. For example, if $\mathcal{P} = \{p, q, r\}$, $\ell = 1$, and $Q = \{p(0), p(1), q(1)\}$, then $\langle \mathbf{X}_p, \mathbf{X}_q, \mathbf{X}_r \rangle = \langle \{0, 1\}, \{1\}, \emptyset \rangle$. Given a formula $\phi(\ldots, \mathcal{X}, \ldots)$ and an interpretation $\mathbf{X} = (k, Q)$ of $\mathcal{X}$, we define the formula $\phi(\ldots, \mathbf{X}, \ldots)$ as follows:

$$x \in \mathbf{X}_p := \bigvee_{j \in \mathbf{X}_p} x = j$$

$$\mathcal{X} = \mathbf{X} := \boldsymbol{n} = k \wedge \bigwedge_{p \in \mathcal{P}} \forall \boldsymbol{x} \colon \boldsymbol{x} < \boldsymbol{n} \to (\boldsymbol{x} \in \mathcal{X}_p \leftrightarrow \boldsymbol{x} \in \mathbf{X}_p)$$

$$\phi(\ldots, \mathbf{X}, \ldots) := \forall \mathcal{X} \colon \mathcal{X} = \mathbf{X} \to \phi(\ldots, \mathcal{X}, \ldots)$$

The CEGAR procedure maintains an (initially empty) set $\mathcal{T}$ of *indexed traps* of $\mathcal{N}(1), \mathcal{N}(2), \ldots, \mathcal{N}(n)$, where an indexed trap is a pair $\mathbf{T} = \langle i, Q \rangle$ such that

---

[1] The CEGAR loop for the non-parametric case could be formulated in SAT and solved using a SAT-solver. However, we formulate it in WS1S, since this allows us to give a uniform description of the non-parametric and the parametric cases.

$1 \leq i \leq n$ and Q is a trap of $\mathcal{N}(i)$. After every update of $\mathcal{T}$ the procedure constructs the formula $SafetyCheck_{\mathcal{T}}$, defined as follows:

$$TrapSet_{\mathcal{T}}(\boldsymbol{n}, \mathcal{X}) := \bigvee_{\mathbf{X} \in \mathcal{T}} \mathcal{X} = \mathbf{X}$$

$$PReach_{\mathcal{T}}(\boldsymbol{n}, \mathcal{M}', \mathcal{M}) := \forall \mathcal{X} : \begin{pmatrix} TrapSet(\boldsymbol{n}, \mathcal{X}) \\ \wedge Marked(\boldsymbol{n}, \mathcal{X}, \mathcal{M}) \end{pmatrix} \rightarrow Marked(\boldsymbol{n}, \mathcal{X}, \mathcal{M}')$$

$$PReach_{\mathcal{T}}(\boldsymbol{n}, \mathcal{M}') := \exists \mathcal{M} : Initial(\boldsymbol{n}, \mathcal{M}) \wedge PReach_{\mathcal{T}}(\boldsymbol{n}, \mathcal{M}', \mathcal{M})$$

$$SafetyCheck_{\mathcal{T}} := \forall \boldsymbol{n} \forall \mathcal{M} : \boldsymbol{n} < n \wedge PReach_{\mathcal{T}}(\boldsymbol{n}, \mathcal{M}) \rightarrow Safe(\boldsymbol{n}, \mathcal{M})$$

Intuitively, $PReach_{\mathcal{T}}(\boldsymbol{n}, \mathcal{M}', \mathcal{M})$ states that according to the set $\mathcal{T}$ of (indexed) traps computed so far, $\mathcal{M}'$ could still be reachable from $\mathcal{M}$, because every trap of $\mathcal{T}$ marked at $\mathcal{M}$ is also marked at $\mathcal{M}'$. Therefore, if $SafetyCheck_{\mathcal{T}}$ holds then $\mathcal{T}$ is already strong enough to show that every reachable marking is safe.

If $\mathcal{T}$ is not strong enough, then the negation of $SafetyCheck_{\mathcal{T}}$ is satisfiable. The WS1S-checker returns a counterexample, i.e., a model $\mathbf{M} = \langle n, M \rangle$ of the formula $PReach_{\mathcal{T}}(\boldsymbol{n}, \mathcal{M}) \wedge \neg Safe(\boldsymbol{n}, \mathcal{M})$. Here M is a marking of $\mathcal{N}(n)$, potentially reachable from an initial marking but not safe. In this case we search for a witness trap $\mathcal{X}$ that is marked at every initial marking, but empty at $\mathbf{M}$, with the help of the formula

$$WTrap_{\mathbf{M}}(\boldsymbol{n}, \mathcal{X}) := \begin{pmatrix} Trap(\boldsymbol{n}, \mathcal{X}) \\ \wedge (\forall \mathcal{M} : Initial(\boldsymbol{n}, \mathcal{M}) \rightarrow Marked(\boldsymbol{n}, \mathcal{X}, \mathcal{M})) \\ \wedge Empty(\boldsymbol{n}, \mathcal{X}, \mathbf{M}) \end{pmatrix}$$

If the formula is satisfiable, then the WS1S-checker returns a model $\mathbf{T} = (n, Q)$. The set Q is then a trap of $\mathcal{N}(n)$. We can now take $\mathcal{T} := \mathcal{T} \cup \{\mathbf{T}\}$, and iterate. Observe that after updating $\mathcal{T}$ the interpretation $\mathbf{M} = \langle n, M \rangle$ is no longer a model of $PReach_{\mathcal{T}}(\boldsymbol{n}, \mathcal{M}) \wedge \neg Safe(\boldsymbol{n}, \mathcal{M})$. Since $\mathcal{N}(1), \ldots, \mathcal{N}(n)$ only have finitely many traps, the procedure eventually terminates.

A CEGAR approach for the parameterized case. In all nontrivial examples, proving safety of the infinitely many instances requires to compute infinitely many traps. Since the previous procedure only computes one trap per iteration, it does not terminate. The way to solve this problem is to insert a *parametrization step* that transforms the witness trap $\mathbf{T} = \langle k, Q \rangle$ into a formula $ParTrap_{\mathbf{T}}(\boldsymbol{n}, \mathcal{X})$ satisfying two properties: (1) all models of the formula are traps, and (2) $\mathbf{T}$ is a model. Since $ParTrap_{\mathbf{T}}(\boldsymbol{n}, \mathcal{X})$ can have infinitely many models, it constitutes a finite representation of an infinite set of traps. These models are also similar to each other and can be understood as capturing a single property of the system.

Example 4. Consider a parameterized net $\mathcal{N} = \langle \mathcal{P}, Tr \rangle$ exhibiting rotational symmetry: For every instance $\mathcal{N}(n)$, a pair (P_1, P_2) of sets is a transition of $\mathcal{N}(n)$ iff the pair $(P_1 \oplus_n 1, P_2 \oplus_n 1)$ is also a transition, where $P \oplus_n 1$ denotes the result of increasing all indices by 1 modulo n. Assume that $\mathcal{P} = \{p, q, r\}$ and

$\mathbf{T} = \langle 3, \{p(1), q(2)\}\rangle$, i.e., $\{p(1), q(2)\}$ is a trap of $\mathcal{N}(3)$. It is intuitively plausible (and we will later prove) that, due to the rotational symmetry, $\{p(i), q(i \oplus_m 1)\}$ is a trap of $\mathcal{N}(j)$ for every $m \geq 3$ and every $0 \leq i \leq m - 1$. We can then define the formula $ParTrap_{\mathbf{X}}(n, \mathcal{X})$ as:

$$ParTrap_{\mathbf{T}}(\boldsymbol{n}, \mathcal{X}) := \boldsymbol{n} \geq 3 \wedge \exists \boldsymbol{i} \colon \boldsymbol{i} < \boldsymbol{n}$$

$$\wedge \, \forall \boldsymbol{x} \colon \boldsymbol{x} < \boldsymbol{n} \to \begin{pmatrix} (\boldsymbol{x} \in \mathcal{X}_p \leftrightarrow \boldsymbol{x} = \boldsymbol{i}) \\ \wedge \, (\boldsymbol{x} \in \mathcal{X}_q \leftrightarrow \boldsymbol{x} = \boldsymbol{i} \oplus_{\boldsymbol{n}} 1) \\ \wedge \, \boldsymbol{x} \notin \mathcal{X}_r \end{pmatrix}.$$

Now, in order to describe the CEGAR procedure for the parameterized case we only need to *redefine* the formula $TrapSet_{\mathcal{T}}(n, \mathcal{X})$. Instead of the formula $TrapSet_{\mathcal{T}}(n, \mathcal{X}) := \bigvee_{\mathbf{T} \in \mathcal{T}} \mathcal{X} = \mathbf{T}$, which holds only when $\mathcal{X}$ is one of the finitely many traps in $\mathcal{T}$, we insert the parametrization procedure and define

$$TrapSet_{\mathcal{T}}(\boldsymbol{n}, \mathcal{X}) := \bigvee_{\mathbf{T} \in \mathcal{T}} ParTrap_{\mathbf{T}}(\boldsymbol{n}, \mathcal{X})$$

All the other formulas remain untouched. The question is how to obtain the formula $ParTrap_{\mathbf{T}}(n, \mathcal{X})$ from $\mathbf{T}$. We discuss this point in the rest of the section.

A semi-automatic approach If we *guess* the formula $ParTrap_{\mathbf{T}}(n, \mathcal{X})$ we can use the WS1S-checker to automatically prove that the guess is correct. Indeed, it suffices to check that all models of $ParTrap_{\mathbf{T}}(n, \mathcal{X})$ are traps, which reduces to proving validity of the formula

$$\forall \boldsymbol{n} \forall \mathcal{X} \colon ParTrap_{\mathbf{T}}(\boldsymbol{n}, \mathcal{X}) \to Trap(\boldsymbol{n}, \mathcal{X})$$

Let us see how this works in Example 1. Assume that the CEGAR procedure produces a trap $\mathbf{T} = \langle 3, \{p(1), q(2)\}\rangle$. The user finds it plausible that, due to the identical behavior of philosophers $1, 2, \ldots, n - 1$, the set $\{p(i), q(i \oplus 1)\}$ will be a trap of $\mathcal{N}(n)$ for every $n \geq 3$ and for every $1 \leq i \leq n - 2$ (i.e., the user excludes the case in which i or $i \oplus_n 1$ are equal to 0). So the user guesses a new formula

$$ParTrap_{\mathbf{T}}(\boldsymbol{n}, \mathcal{X}) := \boldsymbol{n} \geq 3 \wedge \exists \boldsymbol{i} \colon (1 \leq \boldsymbol{i} \leq \boldsymbol{n} - 2) \wedge \forall \boldsymbol{x} \colon$$
$$(\boldsymbol{x} \in \mathcal{X}_p \leftrightarrow \boldsymbol{x} = \boldsymbol{i}) \wedge (\boldsymbol{x} \in \mathcal{X}_q \leftrightarrow \boldsymbol{x} = \boldsymbol{i} \oplus_{\boldsymbol{n}} 1) \wedge \boldsymbol{x} \notin \mathcal{X}_r.$$

The user now automatically checks that all models of $ParTrap_{\mathbf{T}}(n, \mathcal{X})$ are traps. The formula can then be safely added to $TrapSet_{\mathcal{T}}(n, \mathcal{X})$ as a new disjunct.

An automatic approach for specific architectures. Parameterized Petri nets usually have a regular structure. For example, in the readers-writers problem all processes are indistinguishable, and in the philosophers problem, all right-handed processes behave in the same way. In the next sections we show how the structural properties of ring topologies and crowds (two common structure for parameterized systems) can be exploited to automatically compute the formula $ParTrap_{\mathbf{T}}(n, \mathcal{X})$ for each witness trap $\mathbf{T}$.

6 Trap parametrization in rings

Intuitively, a parameterized net $\mathcal{N}$ is a ring if for every transition of every instance $\mathcal{N}(n)$ there is an index $i \in [n]$ and sets $\mathcal{P}_L, \mathcal{P}_R, \mathcal{Q}_L, \mathcal{Q}_r \subseteq \mathcal{P}$ such that the preset of the transition is $(\mathcal{P}_L \times \{i\}) \cup (\mathcal{P}_R \times \{i \oplus_n 1\})$ and the postset is $(\mathcal{Q}_L \times i) \cup (\mathcal{Q}_R \times i \oplus_n 1)$. In other words, every transition involves only two neighbor processes of the ring. In a fully symmetric ring all processes behave identically, while in a headed ring there is one distinguished process, as in Example 1. To ease presentation in this section we only consider fully symmetric rings. The extension to headed rings can be found in the appendix.

The informal statement "all processes behave identically" is captured by requiring the existence of a finite set of *transition patterns* $\langle \mathcal{P}_L, \mathcal{P}_R, \mathcal{Q}_L, \mathcal{Q}_R \rangle$ such that the transitions of $\mathcal{N}(n)$ are the result of "instantiating" each pattern with all pairs i and $i \oplus_n 1$ of consecutive indices.

Definition 3. *A parameterized net $\mathcal{N} = \langle \mathcal{P}, Tr \rangle$ is a fully symmetric ring if there is a finite set of transition patterns of the form $\langle \mathcal{P}_L, \mathcal{P}_R, \mathcal{Q}_L, \mathcal{Q}_R \rangle$, where $\mathcal{P}_L, \mathcal{P}_R, \mathcal{Q}_L, \mathcal{Q}_r \subseteq \mathcal{P}$, such that for every instance $\mathcal{N}(n)$ the following condition holds: $\langle P, Q \rangle$ is a transition of $\mathcal{N}(n)$ iff there is $i \in [n]$ and a pattern such that $P = \mathcal{P}_L \times \{i\} \cup \mathcal{P}_R \times \{i \oplus_n 1\}$ and $Q = \mathcal{Q}_L \times \{i\} \cup \mathcal{Q}_R \times \{i \oplus_n 1\}$.*

It is possible to decide if a given parameterized Petri net is a fully symmetric ring:

Proposition 2. *There is a formula of WS1S such that a parameterized net is a fully symmetric ring iff the formula holds.*

Proof. We introduce an WS1S formula describing symmetric rings in several steps. To avoid making the following formulas resilient against edge cases we assume that any transition formula $Tr(\boldsymbol{n}, \mathcal{X}, \mathcal{Y})$ enforces a minimal size of its models; i.e., $Tr(\boldsymbol{n}, \mathcal{X}, \mathcal{Y}) \models \boldsymbol{n} > 3$. This streamlines the argument and formulas. However, it is straightforward to adapt the formulas to the full generality.

The following formula expresses that for every transition of every instance there is an index i such that all places in the preset and postset of the transition have index i or $i \oplus_{\boldsymbol{n}} 1$. We call i the index of the transition.

$$\varphi := \forall \boldsymbol{n}, \mathcal{X}, \mathcal{Y} \colon Tr(\boldsymbol{n}, \mathcal{X}, \mathcal{Y}) \longrightarrow$$

$$\exists \boldsymbol{i} : \boldsymbol{i} < \boldsymbol{n} \wedge \forall \boldsymbol{x} : \boldsymbol{x} < \boldsymbol{n} \rightarrow \left(\begin{array}{c} \bigvee_{p \in \mathcal{P}} \boldsymbol{x} \in \mathcal{X}_p \vee \boldsymbol{x} \in \mathcal{Y}_p \\ \leftrightarrow [\boldsymbol{x} = \boldsymbol{i} \vee \boldsymbol{x} = \boldsymbol{i} \oplus_{\boldsymbol{n}} 1] \end{array} \right) \quad (6)$$

Now we express that if some instance, say $\mathcal{N}(n)$, contains a transition with index i, then for every other instance, say $\mathcal{N}(m)$, and for every index $0 \le j \le m$,

substituting j for i yields a transition of $\mathcal{N}(m)$:

$$
\begin{aligned}
\psi := \forall \boldsymbol{n}, \boldsymbol{i}, \mathcal{X}, \mathcal{Y}, \boldsymbol{m}, \boldsymbol{j} \colon\ & (\boldsymbol{i} < \boldsymbol{n} \wedge \mathit{Tr}(\boldsymbol{n}, \mathcal{X}, \mathcal{Y}) \wedge \boldsymbol{j} < \boldsymbol{m}) \longrightarrow \\
& \exists \mathcal{X}', \mathcal{Y}' \colon\ \mathit{Tr}(\boldsymbol{m}, \mathcal{X}', \mathcal{Y}') \wedge \\
& \bigwedge_{p \in \mathcal{P}} \left(\begin{array}{l} \boldsymbol{i} \in \mathcal{X}_p \leftrightarrow \boldsymbol{j} \in \mathcal{X}'_p \\ \wedge\, \boldsymbol{i} \in \mathcal{Y}_p \leftrightarrow \boldsymbol{j} \in \mathcal{Y}'_p \\ \wedge\, \boldsymbol{i} \oplus_{\boldsymbol{n}} 1 \in \mathcal{X}_p \leftrightarrow \boldsymbol{j} \oplus_{\boldsymbol{m}} 1 \in \mathcal{X}'_p \\ \wedge\, \boldsymbol{i} \oplus_{\boldsymbol{n}} 1 \in \mathcal{Y}_p \leftrightarrow \boldsymbol{j} \oplus_{\boldsymbol{m}} 1 \in \mathcal{Y}'_p \end{array} \right).
\end{aligned}
\tag{7}
$$

We prove that $\mathcal{N} = \langle \mathcal{P}, \mathit{Tr} \rangle$ is a fully symmetric ring iff its associated formula $\varphi \wedge \psi$ is valid.

First, we show that φ is valid if and only if for all n and every transition $\langle P_1, P_2 \rangle$ of T_n there is an index $0 \le i \le n-1$ such that $P_1 \cup P_2 \subseteq \mathcal{P} \times \{i, i \oplus_n 1\}$.

Assume for all n and every transition $\langle P_1, P_2 \rangle$ of T_n there is an index $0 \le i \le n-1$ such that $P_1 \cup P_2 \subseteq \mathcal{P} \times \{i, i \oplus_n 1\}$. Then for any interpretation μ of $\boldsymbol{n}, \mathcal{X}, \mathcal{Y}$ with $\mu \models \mathit{Tr}(\boldsymbol{n}, \mathcal{X}, \mathcal{Y})$ we have $\langle P, Q \rangle$ and an index $j < \mu(\boldsymbol{n})$ such that $\mu(\mathcal{X}_p) = P \cap \{p\} \times \{j, j \oplus_{\mu(\boldsymbol{n})} 1\}$ and $\mu(\mathcal{Y}_p) = Q \cap \{p\} \times \{j, j \oplus_{\mu(\boldsymbol{n})} 1\}$ for all $p \in \mathcal{P}$.

Consequently,

$$
\mu[\boldsymbol{i} \mapsto i] \models \forall \boldsymbol{x} : \boldsymbol{x} < \boldsymbol{n} \rightarrow \left(\begin{array}{c} \bigvee_{p \in \mathcal{P}} x \in \mathcal{X}_p \vee x \in \mathcal{Y}_p \\ \leftrightarrow [x = i \vee x = i \oplus_{\boldsymbol{n}} 1] \end{array} \right).
$$

Which renders φ valid in general.

On the other hand, if φ is valid careful examining φ gives the desired result: let $\mu \models \mathit{Tr}(\boldsymbol{n}, \mathcal{X}, \mathcal{Y})$. Then fix any $i \in [\mu(\boldsymbol{n})]$ such that

$$
\mu[\boldsymbol{i} \mapsto i] \models \forall \boldsymbol{x} : \boldsymbol{x} < \boldsymbol{n} \rightarrow \left(\begin{array}{c} \bigvee_{p \in \mathcal{P}} x \in \mathcal{X}_p \vee x \in \mathcal{Y}_p \\ \leftrightarrow [x = i \vee x = i \oplus_{\boldsymbol{n}} 1] \end{array} \right).
$$

For the transition $\langle P, Q \rangle$ of μ; i.e., $P = \{\langle p, i \rangle \in \mathcal{P} \times [\mu(\boldsymbol{n})] \mid i \in \mu(\mathcal{X}_p)\}$ and $Q = \{\langle p, i \rangle \in \mathcal{P} \times [\mu(\boldsymbol{n})] \mid i \in \mu(\mathcal{Y}_p)\}$ we see that $P \subseteq \mathcal{P} \times \{i, i \oplus_{\mu(\boldsymbol{n})} 1\}$ and $Q \subseteq \mathcal{P} \times \{i, i \oplus_{\mu(\boldsymbol{n})} 1\}$.

Using this observation we restrict the remaining argument to the case that every transition of $\mathcal{N}(n)$ has an index $i \in [n]$. It remains to show that – under this condition – ψ is valid if and only if $\mathcal{N}$ is a fully symmetric ring: assume $\mathcal{N}$ to be a fully symmetric ring. Let μ be an arbitrary interpretation of $\boldsymbol{n}, \boldsymbol{m}, \mathcal{X}, \mathcal{Y}, \boldsymbol{i}, \boldsymbol{j}$. If $\mu \not\models \boldsymbol{i} < \boldsymbol{n} \wedge \mathit{Tr}(\boldsymbol{n}, \mathcal{X}, \mathcal{Y}) \wedge \boldsymbol{j} < \boldsymbol{m}$ then there is nothing to show. Let now $\mu \models \boldsymbol{i} < \boldsymbol{n} \wedge \mathit{Tr}(\boldsymbol{n}, \mathcal{X}, \mathcal{Y}) \wedge \boldsymbol{j} < \boldsymbol{m}$. Let $n = \mu(\boldsymbol{n})$, $m = \mu(\boldsymbol{m})$, $i = \mu(\boldsymbol{i})$, $j = \mu(\boldsymbol{j})$ and $\langle P, Q \rangle$ such that $P = \{\langle p, i \rangle \in \mathcal{P} \times [\mu(\boldsymbol{n})] \mid i \in \mu(\mathcal{X}_p)\}$ and $Q = \{\langle p, i \rangle \in \mathcal{P} \times [\mu(\boldsymbol{n})] \mid i \in \mu(\mathcal{Y}_p)\}$. Since $\mathcal{N}$ is assumed to be a fully symmetric ring we know that $\langle P, Q \rangle$ is an instance of the pattern $\langle \mathcal{P}_L, \mathcal{P}_R, \mathcal{Q}_L, \mathcal{Q}_R \rangle$ at

an index i. More formally, $P = P_L \times \{i\} \cup P_R \times \{i \oplus_n 1\}$ and $Q = Q_L \times \{i\} \cup Q_R \times \{i \oplus_n 1\}$. If $\mu(i) \notin \{i, i \oplus_n 1\}$, then expanding the interpretation μ to an interpretation μ' which chooses values $\mu'(\mathcal{X}')$ and $\mu'(\mathcal{Y}')$ which yield a transition $\langle P', Q' \rangle$ as an instance of $\langle \mathcal{P}_L, \mathcal{P}_R, \mathcal{Q}_L, \mathcal{Q}_R \rangle$ for an index j' such that $\{j', j' \oplus_m 1\} \cap \{j, j \oplus_m 1\} = \emptyset$. (Note that we use implicitly here that the formula Tr enforces models of sufficient size. Adapting ψ such that i has to be the index of $\langle P, Q \rangle$ is straightforward.)

On the other hand, if $\mu(i) = i$ then expanding μ to μ' with values for $\mu'(\mathcal{X}')$ and $\mu'(\mathcal{Y}')$ such that the associated $\langle P', Q' \rangle$ is an instance of $\langle \mathcal{P}_L, \mathcal{P}_R, \mathcal{Q}_L, \mathcal{Q}_R \rangle$ at index $\mu(j)$ yields the desired result. Analogously, for $\mu(i) = i \oplus_n 1$. It follows that ψ is valid.

Now, assume ψ to be valid. The result follows from carefully examining ψ. For any transition $\langle P, Q \rangle$ in an instance $\mathcal{N}(n)$ we can extract its structure; i.e., a pattern $\langle \mathcal{P}_L, \mathcal{P}_R, \mathcal{Q}_L, \mathcal{Q}_R \rangle$ such that $P = P_L \times \{i\} \cup P_R \times \{i \oplus_n 1\}$ and $Q = Q_L \times \{i\} \cup Q_R \times \{i \oplus_n 1\}$ for an appropriate i (remember that we assume φ to be valid). By the validity of ψ we see that the same pattern can be instantiated (represented by the choice of $\mathcal{X}'$ and $\mathcal{Y}'$) at all other indices (corresponding to the choice for $\mu(j)$) for all other instances (corresponding to the choice for $\mu(m)$).

We need to distinguish between *global* and *local* traps of an instance. Loosely speaking, a global trap contains places of all processes, while a local trap does not. To understand why this is relevant, consider a fully symmetric ring $\mathcal{N} = \langle \mathcal{P}, Tr \rangle$ where $\mathcal{P} = \{p, q\}$ and the transitions of each instance $\mathcal{N}(n)$ are the pairs $\langle \{p(i), q(i \oplus_n 1)\}, \{p(i \oplus_n 1), q(i)\} \rangle$ for every $i \in [n]$. The sets $\{p(0), q(0)\}$ and $\{p(0), p(1), p(2), p(3)\}$ are both traps of $\mathcal{N}(4)$ (they are even 1-balanced sets). However, they are of different nature. Intuitively, in order to decide that $\{p(0), q(0)\}$ is a trap it is not necessary to inspect all of $\mathcal{N}(4)$, but only process 0 and its neighborhood. On the contrary, $\{p(0), \ldots, p(3)\}$ involves all the processes. This has consequences when parameterizing. Due to the symmetry of the ring, $\{p(i), q(i)\}$ is a trap of every instance $\mathcal{N}(n)$ for every $i \in [n]$. However, $\{p(i), p(i \oplus_n 1), \ldots, p(i \oplus_n 3)\}$ is *not* a trap of every instance for every $i \in [n]$, for example $\{p(0), \ldots, p(3)\}$ is not a trap of $\mathcal{N}(5)$. The correct parametrization is a different one, namely $\{p(0), p(1), \ldots, p(n-1)\}$. The difference between the two traps is captured by the following definition.

Definition 4. *Let* $\mathcal{N} = \langle \mathcal{P}, Tr \rangle$ *be a parameterized net. An indexed trap* $\mathbf{T} = \langle n, Q \rangle$ *of* $\mathcal{N}$ is global *if* $Q \cap (\mathcal{P} \times \{i\}) \neq \emptyset$ *for every* $i \in [n]$, *otherwise* $\mathbf{T}$ *is* local.

6.1 Parameterizing local traps

We first observe that local indexed traps can be "shifted" locally while maintaining their trap property.

Lemma 1. *Let* $\mathcal{N} = \langle \mathcal{P}, Tr \rangle$ *be a fully symmetric ring and let* $\langle n, Q \rangle$ *be a local indexed trap of* $\mathcal{N}$. *Then* $\langle n, Q' \rangle$ *with* $Q' = \{\langle p, i \oplus_n 1 \rangle : \langle p, i \rangle \in Q\}$ *is a local indexed trap of* $\mathcal{N}$.

Proof. Assume Q' is not an indexed local trap. Then there is $t \in T_n$ such that ${}^\bullet t \cap Q' \neq \emptyset = t^\bullet \cap Q'$. Since $\mathcal{N}$ is a fully symmetric ring, there is a pattern $\langle\langle P_L, P_R\rangle, \langle Q_L, Q_R\rangle\rangle$ and an index $i \in [n]$ such that t is the instance of $\langle\langle P_L, P_R\rangle, \langle Q_L, Q_R\rangle\rangle$ with index i. Let t' be the transition obtained from the same pattern with index $i \oplus_n (n-1)$; i.e., moved one index to the left. It follows ${}^\bullet t' = \{\langle p, j \oplus_n (n-1)\rangle : \langle p, j\rangle \in {}^\bullet t\}$ and $t'^\bullet = \{\langle p, j \oplus_n (n-1)\rangle : \langle p, j\rangle \in t^\bullet\}$. By definition of Q' we have $Q = \{\langle p, j \oplus_n (n-1)\rangle : \langle p, j\rangle \in Q'\}$. That, however, gives ${}^\bullet t' \cap Q \neq \emptyset = t'^\bullet \cap Q$ in contradiction to Q being a local indexed trap. $\square$

Our second lemma states that for any indexed local traps $\langle n, Q\rangle$ with $Q \cap (\mathcal{P} \times \{n-1\})$, the set Q remains a trap in any instance $\mathcal{N}(n')$ with $n \leq n'$.

Lemma 2. *Let $\mathcal{N}$ be a fully symmetric ring and $\langle n, Q\rangle$ a local indexed trap with $Q \cap (\mathcal{P} \times \{n-1\}) = \emptyset$. Then $\langle n', Q\rangle$ is a local indexed trap for all $n' \geq n$.*

Proof. Assume the statement is false. That is, $\langle n', Q\rangle$ is not a local indexed trap. If $n' = n$ there is an immediate contradiction with the assumption that $\langle n, Q\rangle$ is a local indexed trap. Hence, let $n' > n$ *minimal* such that $\langle n', Q\rangle$ is *not* a local indexed trap. So there is a transition t in $\mathcal{N}(n')$ such that ${}^\bullet t \cap Q \neq \emptyset = t^\bullet \cap Q$. Since $Q \cap (\mathcal{P} \times \{n-1\}) = \emptyset$ by assumption of the lemma and $n' > n$ by case distinction we have $Q \cap (\mathcal{P} \times \{n'-2, n'-1\}) = \emptyset$. With this and the facts that fully symmetric rings only allow for transitions using places of two adjacent indices and ${}^\bullet t \cap Q \neq \emptyset$ we get ${}^\bullet t \cap \mathcal{P} \times \{n'-1\} = \emptyset$ and $t^\bullet \cap \mathcal{P} \times \{n'-1\} = \emptyset$. That means, however, that t is also a transition in $\mathcal{N}(n'-1)$ because $\mathcal{N}$ is a fully symmetric ring and, consequently, $\langle n'-1, Q\rangle$ already is not a local indexed trap. This contradicts that n' was chosen minimal and concludes the proof. $\square$

We can now show how to obtain a sound parameterization of a given indexed trap. The formula $ParTrap_{\mathbf{T}}(\mathcal{X})$ states that $\mathcal{X}$ is the result of "shifting" $\mathbf{T} = \langle n, Q\rangle$ in $\mathcal{N}(n')$ for some $n' \geq n$.

Theorem 1. *Let $\mathcal{N} = \langle \mathcal{P}, Tr\rangle$ be a fully symmetric ring and let $\langle n, Q\rangle$ be a local indexed trap of $\mathcal{N}(n)$ such that $Q \subseteq (\mathcal{P} \times I)$ for a minimal set $I \subset [n]$. Assume $I = \{i_0, \ldots, i_{k-1}\}$ with $0 \leq i_0 < i_1 < \ldots < i_{k-1} < n-1$. Then every model of the formula*

$$ParTrap_{\mathbf{T}}(\boldsymbol{n}, \mathcal{X}) := n \leq \boldsymbol{n} \ \wedge \ \exists \boldsymbol{y} \colon \boldsymbol{y} < \boldsymbol{n} \ \wedge \ \bigwedge_{p \in \mathcal{P}} \forall \boldsymbol{x} \colon \boldsymbol{x} < \boldsymbol{n} \rightarrow$$

$$\left(\boldsymbol{x} \in \mathcal{X}_p \leftrightarrow \left(\begin{array}{c} \bigvee_{\langle i_0, p\rangle \in Q} \boldsymbol{x} = \boldsymbol{y} \\ \vee \bigvee_{j > 0, \langle i_j, p\rangle \in Q} \boldsymbol{x} = \boldsymbol{y} \oplus_{\boldsymbol{n}} (i_j - i_{j-1}) \end{array} \right) \right)$$

is an indexed trap of $\mathcal{N}$.

Proof. Assume $\mu \models ParTrap_{\mathbf{T}}(\boldsymbol{n}, \mathcal{X})$. Then there exists a tuple $\langle k, P\rangle$ such that $\mu(\boldsymbol{n}) = k$ and $\mu(\mathcal{X}_p) = \{i \in [k] \mid \langle p, i\rangle \in P\}$ for every $p \in \mathcal{P}$. We have $n \leq k$ by

the first conjunct of $ParTrap_{\mathbf{T}}$. Let $j \in [k]$ be any value such that assigning j to $\mathbf{y}$ satisfies the existentially quantified subformula of $ParTrap_{\mathbf{T}}$.

Since $i_{k-1} < n-1$ we have $Q \cap (\mathcal{P} \times \{n-1\}) = \emptyset$. So we can apply Lemma 2 to $\langle n, Q \rangle$, and in fact we can apply it $(n-k)$ times, yielding a local trap $\langle k, Q \rangle$. Now, fix indices $i'_0, \ldots, i'_{k-1}$ such that $i'_0 = j$ and $i'_{\ell+1} = i'_\ell \oplus_k (i_{\ell+1} - i_\ell)$ for $0 \le \ell < k-1$. Carefully examining $ParTrap_{\mathbf{T}}$ one can now observe that

$$\{p \in \mathcal{P} \mid \langle p, i'_\ell \rangle \in P\} = \{p \in \mathcal{P} \mid \langle p, i_\ell \rangle \in P\} \text{ for all } 0 \le \ell < k$$

and $\emptyset = \mathcal{P} \times ([k] \setminus \{i'_0, \ldots, i'_{k-1}\})$. We can now apply $(k-i_0)+i'_0$ times Lemma 1 to the local trap $\langle k, Q \rangle$, which shows that $\langle k, P \rangle$ is a local trap.

Remark 1. Since Theorem 1 requires $i_{k-1} < n-1$, it can only be applied to local traps $\langle n, Q \rangle$ such that $Q \cap (\mathcal{P} \times \{n-1\}) = \emptyset$. However, for every local trap $\langle n, Q \rangle$ Lemma 1 allows us to find a local trap $\langle n, Q' \rangle$ satisfying $Q' \cap (\mathcal{P} \times \{n-1\}) = \emptyset$, which we can then parameterize applying Theorem 1.

6.2 Parameterizing global traps

In contrast to local traps, global traps involve all indices $[n]$ of the instance $\mathcal{N}(n)$. Let $\langle n, Q \rangle$ be an indexed global trap. We denote with $Q[i]$ the set $P \subseteq \mathcal{P}$ such that $P \times \{i\} = Q \cap (\mathcal{P} \times \{i\})$; i.e., the set of places in Q at index i. Moreover, we say Q has period p if p is the smallest divisor of n such that for all $0 \le j < p$ we have $Q[j] = Q[k \cdot p + j]$ for all $0 \le k < \frac{n}{p}$. That is, Q is a repetition of the same p sets in a row. Since n is a period of Q we know that every Q has a period, which we denote p_Q. Recall the global trap $Q = \{p(0), p(1), p(2), p(3)\}$ from before. Then, $Q[0] = Q[1] = Q[2] = Q[3] = \{p\}$ and, consequently, $p_Q = 1$. Intuitively, we can repeat a period over and over again and still obtain a trap. So we can parameterize global traps by capturing the repetition of periodic behavior:

Theorem 2. *Let $\langle n, Q \rangle$ be an indexed global trap with $n \ge 2$. Then every model of the formula*

$$ParTrap_{\mathbf{T}}(\mathbf{n}, \mathcal{X}) := \exists \mathbf{P} : 0 \in \mathbf{P} \wedge \mathbf{n} \in \mathbf{P}$$

$$\wedge\ \forall \mathbf{x} : \mathbf{x} \le \mathbf{n} \to \mathbf{x} \in \mathbf{P} \leftrightarrow \left(\begin{array}{c} \bigwedge\limits_{0 \le k < p_Q} \mathbf{x} + k \notin \mathbf{P} \\ \wedge\ \mathbf{x} + p_Q \in \mathbf{P} \end{array} \right)$$

$$\wedge\ \forall \mathbf{x}_0, \ldots, \mathbf{x}_{p_Q-1} : \left(\begin{array}{c} \bigwedge\limits_{0 < k \le p_Q-1} \mathbf{x}_{k-1} + 1 = \mathbf{x}_k \\ \wedge\ \mathbf{x}_{p_Q-1} < \mathbf{n} \wedge \mathbf{x}_0 \in \mathbf{P} \end{array} \right)$$

$$\to \bigwedge\limits_{0 \le k < p_Q} \bigwedge\limits_{p \in Q[k]} \mathbf{x}_k \in \mathcal{X}_p \wedge \bigwedge\limits_{p \in \mathcal{P} \setminus Q[k]} \mathbf{x}_k \notin \mathcal{X}_p$$

is an indexed global trap.

Proof. Let μ be a model of $ParTrap_{\mathbf{T}}(\boldsymbol{n}, \mathcal{X})$. Observe that we have $\mu(\boldsymbol{P}) = \{0, p_Q, 2 \cdot p_Q, \ldots, \ell \cdot p_Q\}$ for some $\ell > 0$. Let $k := \ell \cdot p_Q = \mu(\boldsymbol{n})$ and let P be the set of places of $\mathcal{N}(k)$ such that $\mu(\mathcal{X}_p) = \{i \in [k] : \langle p, i \rangle \in P\}$. Examining $ParTrap_{\mathbf{T}}(\boldsymbol{n}, \mathcal{X})$ further we observe that $P[j \cdot p_Q + o] = Q[o]$ holds for all $0 \leq o < p_Q$ and $0 \leq j < \ell$.

It remains to show that P is indeed a trap. Assume the contrary. Then there is a transition t in $\mathcal{N}(k)$ such that $P \cap {}^\bullet t \neq \emptyset = P \cap t^\bullet$. Since $\mathcal{N}$ is a fully symmetric ring there is an index $i \in [k]$ such that ${}^\bullet t \cup t^\bullet \subseteq \mathcal{P} \times \{i, i \oplus_k 1\}$. Pick j such that $i = j \cdot p_Q + o$ for $0 \leq o < p_Q$. Observe that $P[i] = Q[o]$ and $P[i \oplus_k 1] = Q[o \oplus_n 1]$. Again, by $\mathcal{N}$ being a symmetric ring, we can find a transition t' such that ${}^\bullet t' = \{\langle p, o \rangle : \langle p, i \rangle \in {}^\bullet t\} \cup \{\langle p, o \oplus_n 1 \rangle : \langle p, i \oplus_k 1 \rangle \in {}^\bullet t\}$ and $t'^\bullet = \{\langle p, o \rangle : \langle p, i \rangle \in t^\bullet\} \cup \{\langle p, o \oplus_n 1 \rangle : \langle p, i \oplus_k 1 \rangle \in t^\bullet\}$. This, however, yields a contradiction since t' is a witness for Q not being a trap in contradiction to the assumptions.

7 Trap parametrization in barrier crowds

Barrier crowds are parameterized systems in which communication happens by means of global steps in which each process makes a move. An initiator process decides to start a step, and all the other processes get a chance to veto it; if the step is not blocked (if all the processes accept it), all the processes, including the initiator, update their local state. Barrier crowds are slightly more general than broadcast protocols [26], which, loosely speaking, correspond to the special case in which no process makes use of the veto capability. Like broadcast protocols, barrier crowds can be used to model cache coherence protocols [19].

As for fully symmetric rings, transitions of the instances of a barrier crowd are generated from a finite set of "transition patterns". A transition pattern of a barrier crowd $\mathcal{N}$ is a pair $\langle \mathcal{I}, \mathbb{A} \rangle$, where $\mathcal{I} \in 2^{\mathcal{P}} \times 2^{\mathcal{P}}$ and $\mathbb{A} \subseteq 2^{\mathcal{P}} \times 2^{\mathcal{P}}$. Assume for example that each process can be in states p, q, r, and maintains a boolean variable with values $\{0, 1\}$. The corresponding parameterized net has $\mathcal{P} = \{p, q, r, 0, 1\}$ as set of places. Consider the transition pattern with $\mathcal{I} = \langle \{p, 0\}, \{q, 1\} \rangle$, and $\mathbb{A} = \{\langle \{p\}, \{p\} \rangle, \langle \{q, 0\}, \{r, 0\} \rangle, \langle \{q, 1\}, \{r, 0\} \rangle\}$. This pattern models that the initiator process, say process i, proposes a step that takes it from p to q, setting its variable to 1. Each other process reacts as follows, depending on its current state: if in p, it stays in p, leaving the variable unchanged; if in q, it moves to r, setting the variable to 0; if in r, it vetoes the step (because $\mathbb{A}$ does not offer a way to accept from state r).

Definition 5. *A parameterized Petri net $\mathcal{N} = \langle \mathcal{P}, Tr \rangle$ is a barrier crowd if there is a finite set of transition patterns of the form $\langle \mathcal{I}, \mathbb{A} \rangle$ such that for every instance $\mathcal{N}(n)$ the following condition holds: a pair $\langle P, Q \rangle$ is a transition of $\mathcal{N}(n)$ iff there exists a pattern $\langle \mathcal{I}, \mathbb{A} \rangle$ and $i \in [n]$ such that:*

- *$P \cap (\mathcal{P} \times \{i\}) = P_I \times \{i\}$ and $Q \cap (\mathcal{P} \times \{i\}) = Q_I \times \{i\}$, where $\mathcal{I} = \langle P_I, Q_I \rangle$.*
- *for every $j \neq i$ there is $\langle P_A, Q_A \rangle \in \mathbb{A}$ such that $P \cap (\mathcal{P} \times \{j\}) = P_A \times \{j\}$ and $Q \cap (\mathcal{P} \times \{j\}) = Q_A \times \{j\}$.*

Note that the number of transitions of $\mathcal{N}(n)$ grows quickly in n, even though the structure of the system remains simple, making parameterized verification particularly attractive.

In the rest of the section we present an automatic parametrization procedure for traps of barrier crowds. First we show that barrier crowds satisfy two important structural properties.

Given a set of places $P \subseteq \mathcal{P} \times [n]$ and a permutation $\pi\colon [n] \to [n]$, let $\pi(P)$ denote the set of places $\{p(\pi(i)) : p(i) \in P\}$. Given an index $0 \leq k < n$, let $\mathrm{drop}_{k,n}(P)$ denote the set of places defined as follows: $p(i) \in \mathrm{drop}_{k,n}(P)$ iff either $0 \leq i < k$ and $p(i) \in P$, or $k < i \leq n-1$ and $p(i+1) \in P$.

Definition 6. *Let $\mathcal{N}$ be a parameterized Petri net. A transition $\langle P_1, P_2 \rangle$ of $\mathcal{N}(n)$ is:*

- *order invariant if $\langle \pi(P_1), \pi(P_2) \rangle$ is also a transition of $\mathcal{N}(n)$ for every permutation $\pi\colon [n] \to [n]$.*
- *homogeneous if there is an index $0 \leq i < n$ such that for every $k \in [n] \setminus \{i\}$ the pair $\langle \mathrm{drop}_{k,n}(P_1), \mathrm{drop}_{k,n}(P_2) \rangle$ is a transition of $\mathcal{N}(n-1)$.*

$\mathcal{N}$ is homogeneous (order invariant) if all transitions of all instances $\mathcal{N}(n)$ is homogeneous (order invariant).

Intuitively, order invariance indicates that processes are indistinguishable. Homogeneity indicates that transitions in the large instances are not substantially different from the transitions in the smaller ones.

Proposition 3. *Barrier crowds are order invariant and homogeneous.*

Proof. Let $\mathcal{N}$ be a barrier crowd. For order invariance, let $\langle P, Q \rangle$ be a transition of an instance $\mathcal{N}(n)$, and let $\pi\colon [n] \to [n]$ be a permutation. We show that $\langle \pi(P), \pi(Q) \rangle$ is also a transition of $\mathcal{N}(n)$. By the definition of barrier crowds there is a pattern $\langle \mathcal{I}, \mathbb{A} \rangle$, where $\mathcal{I} = \langle P_I, Q_I \rangle$, and an index i such that

- $P \cap (\mathcal{P} \times \{i\}) = P_I \times \{i\}$ and $Q \cap (\mathcal{P} \times \{i\}) = Q_I \times \{i\}$; and
- for every $j \neq i$ there is $\left\langle P_A^j, Q_A^j \right\rangle \in \mathbb{A}$ such that $P \cap (\mathcal{P} \times \{j\}) = P_A^j \times \{j\}$ and $Q \cap (\mathcal{P} \times \{j\}) = Q_A^j \times \{j\}$.

Intuitively, by the definition of barrier crowds, the result of instantiating $\langle \mathcal{I}, \mathbb{A} \rangle$ with the index $\pi(i)$ instead of i is also a transition of $\mathcal{N}(n)$. Formally, the pair $\langle P', Q' \rangle$ given by

- $P' \cap (\mathcal{P} \times \{\pi(i)\}) = P_I \times \{\pi(i)\}$ and $Q' \cap (\mathcal{P} \times \{\pi(i)\}) = Q_I \times \{\pi(i)\}$, and
- $P' \cap (\mathcal{P} \times \{\pi(j)\}) = P_A^j \times \{\pi(j)\}$ and $Q' \cap (\mathcal{P} \times \{\pi(j)\}) = Q_A^j \times \{\pi(j)\}$ for every $j \neq i$

is a transition of $\mathcal{N}(n)$. By construction we have $\pi(P) = P'$ and $\pi(Q) = Q'$. So $\langle \pi(P), \pi(Q) \rangle$ is a transition of $\mathcal{N}(n)$, and we are done.

For homogeneity, let $\langle P, Q \rangle$ be a transition of $\mathcal{N}(n)$. Let $\langle \mathcal{I}, \mathbb{A} \rangle$ with $\mathcal{I} = \langle P_I, Q_I \rangle$ be the pattern of which $\langle P, Q \rangle$ is an instance, i.e., $P \cap (\mathcal{P} \times \{i\}) = P_I \times \{i\}$

and $Q \cap (\mathcal{P} \times \{i\}) = Q_I \times \{i\}$ for every $i \in [n]$. By the definition of barrier crowds, for every $j \in [n] \setminus \{i\}$ there is $\left\langle P_I^j, Q_I^j \right\rangle \in \mathbb{A}$ such that $P \cap (\mathcal{P} \times \{j\}) = P_I^j$ and $Q \cap (\mathcal{P} \times \{j\}) = Q_I^j$. For every $k \in [n] \setminus \{i\}$, we carefully instantiate $\langle \mathcal{I}, \mathbb{A} \rangle$ to obtain a transition $\langle P', Q' \rangle$ satisfying $\langle P', Q' \rangle = \left\langle \mathrm{drop}_{k,n}(P), \mathrm{drop}_{k,n}(Q) \right\rangle$, which concludes the proof. We need to consider two cases:

- If $i < k$, then:
  - $P' \cap (\mathcal{P} \times \{i\}) = P_I$ and $Q' \cap (\mathcal{P} \times \{j\}) = P_I^j$ for all $i \neq j < k$;
  - $P' \cap (\mathcal{P} \times \{j\}) = P_I^j$ and $P' \cap (\mathcal{P} \times \{j\}) = P_I^{j-1}$ for all $k < j$;
  - $Q' \cap (\mathcal{P} \times \{i\}) = Q_I$ and $Q' \cap (\mathcal{P} \times \{j\}) = Q_I^j$ for all $i \neq j < k$; and
  - $Q' \cap (\mathcal{P} \times \{j\}) = Q_I^j$ and $Q' \cap (\mathcal{P} \times \{j\}) = Q_I^{j-1}$ for all $k < j$.
- If $k < i$, then $\langle P', Q' \rangle$ is defined as for the case $i < k$, with the exception that now $P' \cap (\mathcal{P} \times \{i-1\}) = P_I$ and $Q' \cap \mathcal{P} \times \{i-1\} = Q_I$.

In both cases $\langle P', Q' \rangle$ is a transition in $\mathcal{N}(n-1)$ by definition of barrier crowds. $P' = \mathrm{drop}_{k,n}(P)$ and $Q' = \mathrm{drop}_{k,n}(Q)$ which concludes the argument.

7.1 Parameterizing traps for barrier crowds

By order invariance, if Q is a trap of an instance, say $\mathcal{N}(n)$, then $\pi(Q)$ is also a trap for every permutation π. The set of all traps that can be obtained from Q by permutations can be described as a multiset $\mathcal{Q} \colon 2^{\mathcal{P}} \to [n]$. For example, assume $\mathcal{P} = \{p, q\}$, $n = 5$, and $Q = \{p(0), p(1), q(1), p(2), q(2), q(4)\}$. Then $\mathcal{Q}(\{p, q\}) = 2$ (because of indices 1 and 2), $\mathcal{Q}(\{p\}) = \mathcal{Q}(\{q\}) = 1$ (index 0 and 4, respectively), and $\mathcal{Q}(\emptyset) = 1$ (index 3). Any assignment of indices to the elements of $\mathcal{Q}$ results in a trap. We call $\mathcal{Q}$ the *trap family of Q*.

Proposition 4. *Let $\mathcal{N}$ be an order invariant and homogeneous parameterized Petri net, let Q be a trap of an instance $\mathcal{N}(n)$, and let $\mathcal{Q} \colon 2^{\mathcal{P}} \to [n]$ be the trap family of Q. We have:*

- *If $\mathcal{Q}(\emptyset) \geq 1$ and $\mathcal{Q}'$ is obtained from $\mathcal{Q}$ by increasing the multiplicity of $\emptyset$, then $\mathcal{Q}'$ is also a trap family of another instance of $\mathcal{N}$.*
- *For every $S \in 2^{\mathcal{P}}$, if $\mathcal{Q}(S) \geq 2$ and $\mathcal{Q}'$ is obtained from $\mathcal{Q}$ by increasing the multiplicity of S, then $\mathcal{Q}'$ is also a trap family of another instance of $\mathcal{N}$.*

Proof. First consider increasing the multiplicity of $\emptyset$. It suffices to consider the case of the family $\mathcal{Q}'$ obtained from $\mathcal{Q}$ by setting $\mathcal{Q}'(\emptyset) = \mathcal{Q}(\emptyset) + 1$, since the general statement follows by induction in a straightforward manner. Assume that $\langle n+1, \mathcal{Q}' \rangle$ is not a trap in $\mathcal{N}(n+1)$, but has the multiplicities of $\mathcal{Q}'$. Let t' be a transition of $\mathcal{N}(n+1)$ such that $Q' \cap {}^\bullet t' \neq \emptyset = Q' \cap t'^\bullet$. Since $\mathcal{Q}'(\emptyset) \geq 2$, there are at least two distinct indices i, k such that $Q' \cap (\mathcal{P} \times \{i, k\}) = \emptyset$. By homogeneity of $\mathcal{N}$ we can choose i, k so that a transition t in $\mathcal{N}(n)$ satisfies ${}^\bullet t = \mathrm{drop}_{k,n}({}^\bullet t')$ and $t^\bullet = \mathrm{drop}_{k,n}(t'^\bullet)$. Further, let $Q = \mathrm{drop}_{k,n}(Q')$. Note that Q is an instance of the trap family $\mathcal{Q}$. However, ${}^\bullet t \cap Q \neq \emptyset = t^\bullet \cap Q$ by the definition of t and $\mathrm{drop}_{k,n}$ in contradiction to $\mathcal{Q}$ being a trap family.

In the case of increasing the multiplicity of a non-empty set S we know that $\mathcal{Q}(S) \geq 2$ and $\mathcal{Q}'(S) = \mathcal{Q}(S)+1 \geq 3$. The argument is analogous to the previous case. First we assume $\langle n+1, Q' \rangle$ is an instance of $\mathcal{Q}'$ that is not a trap. For Q', let k_1, k_2, k_3 be three distinct indices in $[n+1]$ such that $Q' \cap \mathcal{P} \times \{k_1, k_2, k_3\} = S \times \{k_1, k_2, k_3\}$. Then, we find a transition t' in $\mathcal{N}(n+1)$ witnessing that Q' is not a trap. We consider wo cases:

- $\bullet t' \cap S \times \{k_1, k_2, k_3\} = \emptyset$.
  By homogeneity, there is $k \in \{k_1, k_2, k_3\}$ such that the result of applying the $\mathrm{drop}_{k,n}$ operation to t' is a transition t of $\mathcal{N}(n)$. The set $Q = \mathrm{drop}_{k,n}(Q')$ is a set of places of $\mathcal{N}(n)$ with trap family $\mathcal{Q}$. We have $\bullet t \cap Q \neq \emptyset$ since $\bullet t' \cap Q' \neq \emptyset$ and $Q' \cap (\mathcal{P} \times \{k\}) = \emptyset$; further, $t^\bullet \cap Q = \emptyset$ since $t'^\bullet \cap Q' = \emptyset$. So Q is not a trap, contradicting the assumption.

- $\bullet t' \cap S \times \{k_1, k_2, k_3\} \neq \emptyset$.
  By homogeneity there are $k_1', k_2' \in \{k_1, k_2, k_3\}$ such that the result of applying $\mathrm{drop}_{k_1',n}$ and $\mathrm{drop}_{k_2',n}$ to t' are two transitions t_1 and t_2 of $\mathcal{N}(n)$. Since $t'^\bullet \cap Q' = \emptyset$ we also have $t_1^\bullet \cap Q = \emptyset$ and $t_1^\bullet \cap Q = \emptyset$. Let k_o denote the only element in $\{k_1, k_2, k_3\} \setminus \{k_1', k_2'\}$. If $S \times \{k_o\} \cap \bullet t' \neq \emptyset$ then $\bullet t_1 \cap Q \neq \emptyset$ and $\bullet t_1 \cap Q \neq \emptyset$ because k_o is not dropped. If, on the other hand, $S \times \{k_o\} \cap \bullet t' = \emptyset$, then either $S \times \{k_1\} \cap \bullet t' \neq \emptyset$ or $S \times \{k_2\} \cap \bullet t' \neq \emptyset$. By symmetry we can assume w.l.o.g. $S \times \{k_1\} \cap \bullet t' \neq \emptyset$. Then $\bullet t_2 \cap Q \neq \emptyset$. So Q is not a trap although its trap family is $\mathcal{Q}$, which contradicts the assumption.

Proposition 4 leads to a parameterization procedure for barrier crowds. Given a trap Q of some instance $\mathcal{N}(n)$ and its trap family $\mathcal{Q}$, consider all multisets obtained from $\mathcal{Q}$ by applying the operations of Proposition 4. We call this set of multisets the *extended trap family* of Q. Observe that $\mathcal{Q}$ represents a set of traps of $\mathcal{N}(n)$, while the extended family represents a set of traps across all instances $\mathcal{N}(n')$ with $n' \geq n$.

Give an indexed trap $\mathbf{T} = \langle n, Q \rangle$, we choose the formula $ParTrap_{\mathbf{T}}(\mathcal{X})$ so that its models correspond to the traps of the extended family of Q. For this, we capture the minimal required multiplicities of $\langle n, Q \rangle$ by quantifying for every $S \subseteq \mathcal{P}$ with $\mathcal{Q}(S) > 0$ indices $\boldsymbol{i}_{S,1}, \ldots, \boldsymbol{i}_{S,\mathcal{Q}(S)}$ for which precisely the places in S are marked. Making all indices introduced this way pairwise distinct ensures that any model of the formula at least covers the multiset $\mathcal{Q}$. Additionally, we can capture that the subset S of $\mathcal{P}$ which are marked in every other index are chosen such that Proposition 4 ensures that we still obtain a trap.

$$ParTrap_{\mathbf{T}}(\boldsymbol{n}, \mathcal{X}) := \exists_{S \subseteq \mathcal{P}} \boldsymbol{i}_{S,1}, \ldots, \boldsymbol{i}_{S,\mathcal{Q}(S)} : \left\{ \left(\bigwedge_{(S,k) \neq (S',k')} (\boldsymbol{i}_{S,k} \neq \boldsymbol{i}_{S',k'}) \right) \wedge \right.$$

$$\forall \boldsymbol{j} : \boldsymbol{j} < \boldsymbol{n} \rightarrow \left[\left(\bigvee_{S \subseteq \mathcal{P}, k=1,\ldots,\mathcal{Q}(S)} \left(\boldsymbol{j} = \boldsymbol{i}_{S,k} \wedge \left(\begin{array}{c} \bigwedge_{p \in S} \boldsymbol{j} \in \mathcal{X}_p \\ \wedge \bigwedge_{p \in \mathcal{P} \setminus S} \boldsymbol{j} \notin \mathcal{X}_p \end{array} \right) \right) \right) \vee \right.$$

$$\left. \left. \left(\left(\bigwedge_{S \subseteq \mathcal{P}, k=1,\ldots,\mathcal{Q}(S)} \boldsymbol{j} \neq \boldsymbol{i}_{S,k} \right) \wedge \bigvee_{\substack{\emptyset \neq S \subseteq \mathcal{P} : \, \mathcal{Q}(S) \geq 2 \\ S = \emptyset : \, \mathcal{Q}(S) \geq 1}} \left(\begin{array}{c} \bigwedge_{p \in S} \boldsymbol{j} \in \mathcal{X}_p \\ \wedge \bigwedge_{p \in \mathcal{P} \setminus S} \boldsymbol{j} \notin \mathcal{X}_p \end{array} \right) \right) \right] \right\}.$$

We immediately get:

Theorem 3. *Let $\mathcal{N} = \langle \mathcal{P}, Tr \rangle$ be a barrier crowd and let $\langle n, Q \rangle$ be a local indexed trap of $\mathcal{N}(n)$. Then every model of the formula $ParTrap_{\mathbf{T}}(\boldsymbol{n}, \mathcal{X})$ defined above is an indexed trap of $\mathcal{N}$.*

Remark 2. This theorem applies to all order invariant and homogeneous systems. It is easy to see that order invariance and homogeneity of a given parameterized net can be expressed in WS1S and verified automatically.

8 Experiments

We implemented the CEGAR loop and the parameterization techniques of Sections 6 and 7 in our tool `ostrich`. `ostrich` heavily relies on MONA as a WS1S-solver. The results of our experiments are presented in Figure 2. In the first two columns the table reports the topology and the name of the system to be verified. The array topology is a linear topology where agents can refer existentially or universally to agents with smaller or larger indices. Analogously to the other topologies we derive a sound parameterization technique for traps, 1-BB sets, and siphons. The rings are Dijsktra's token ring for mutual exclusion [34] and a model of the dining philosophers in which philosophers pick both forks simultaneously. For headed rings we consider Example 1 and a model of a message passing leader election algorithm. The array is Burns' mutual exclusion algorithm [39]. The crowds are Dijkstra's algorithm for mutual exclusion [21] and models of cache-coherence protocols taken from [19]. Note that we check inductiveness of the property; i.e., if it holds initially and there is no marking satisfying the property and the current abstraction and reaching in a single step a marking which violates the property. Additionally, we include in the specification of the parameterized Petri net a partition of the places $\mathcal{P}$ such that the places of every index in every instance form a 1BB-set. Collectively, this ensures

that all examples are 1-bounded and yields invariants similar to (1), (2) for Example 1. Since `ostrich` does not compute but only checks these invariants we do not count them in Figure 2 (leading to 3 semi-automatic invariants for Example 1 since we omit (1), and (2)). Moreover, these invariants already imply inductiveness of some safety properties; prominently deadlock-freedom for all considered cache-coherence protocols.

The third column gives the time `ostrich` needs to initialize the analysis; this includes verifying that the given parameterized Petri net is covered by 1BB-sets, and that it indeed has the given topology. The fourth column gives the property being checked. The specification of the cache coherence protocols consists of a number of consistency properties, specific for each protocol. The legend "consistency (x/y)" indicates that the specification consists of y properties, of which `ostrich` was able to automatically prove the inductiveness of x. Column 5 gives the time need to check the inductiveness the property (or, in the case of the cache-coherence protocols, either find a marking which satisfies all constraints imposed by 1BB-sets, traps or siphons, or prove the inductiveness of the properties together). Columns 6, 7, and 8 give the number of WS1S-formulas, each corresponding to a parameterized 1BB-sets, trap, or siphon that are computed by the CEGAR loop. Some of these WS1S-formulas have only one model, i.e., they correspond to a single trap, siphon, or 1BB-set of one instance. Such "artifacts" are needed when small instances (e.g., arrays of size 2) require ad-hoc proofs that cannot be parameterized. In these cases the "real" number of parametric invariants is the result of subtracting the number of artifacts from the total number. The last column reports the number of parameterized inductive invariants obtained by the semi-automatic CEGAR loop. There the user is presented a series of counter examples to the inductiveness of the property. The user can check for traps, siphons or 1BB-sets to disprove the counter example. If the user then provides an invariant which proves inductive it is used to refine the abstraction until no further counter example can be found. The response time of `ostrich` in this setting is immediate which provides a nice user experience. Dragon and MOESI are examples showing that the semi-automatic procedure can lead to proofs with fewer invariants. The last step of the automatic procedure is to remove invariants until no invariant can be removed without obtaining a counter example again.

For Example 1 `ostrich` automatically computes the following family of 1BB-sets (additionally to the invariants (1) and (2)): (For readability we omit some artifacts.)

$$2 \leq \boldsymbol{n} \wedge \text{taken} = \text{think} = \emptyset \wedge \text{wait} = \text{eat} = \{0,1\} \wedge \text{free} = \{1\}$$

$$3 \leq \boldsymbol{n} \wedge \text{taken} = \text{wait} = \text{think} = \emptyset \wedge \text{eat} = \{\boldsymbol{n}-1, 0\} \wedge \text{free} = \{0\}$$

$$4 \leq \boldsymbol{n} \wedge \text{taken} = \text{think} = \emptyset \wedge \text{free} = \text{wait} = \{\boldsymbol{n}-1\} \wedge \text{eat} = \{\boldsymbol{n}-2, \boldsymbol{n}-1\}$$

$$2 \leq \boldsymbol{n} \wedge \exists \boldsymbol{i} : 1 < \boldsymbol{i} < \boldsymbol{n}-2 \wedge \begin{pmatrix} \text{taken} = \text{think} = \emptyset \wedge \text{free} = \text{wait} = \{\boldsymbol{i} \oplus_{\boldsymbol{n}} 1\} \\ \wedge \text{eat} = \{\boldsymbol{i}, \boldsymbol{i} \oplus_{\boldsymbol{n}} 1\} \end{pmatrix}$$

Fig. 2. Experimental results of `ostrich`. The complete data is available at [45].

Topology	Example	Init. (ms)	Property	Check (ms)	1BB-sets	Traps	Siphons	Semi-automatic invariants
ring	Dijkstra ring	558	deadlock	40	1 (1)	0 (0)	0 (0)	2
			mutual exclusion	125	1 (1)	1 (1)	0 (0)	
	atomic phil.	409	deadlock	79	1 (1)	0 (0)	0 (0)	4
headed ring	lefty phil.	495	deadlock	294	7 (4)	0 (0)	0 (0)	3
	leader election	670	not 0 and $n-1$ leader	965	1 (0)	0 (0)	2 (1)	1
			not two leaders	–	–	–	–	
array	Burns	501	deadlock	16	0 (0)	0 (0)	0 (0)	1
			mutual exclusion	379	0 (0)	8 (7)	0 (0)	
crowd	Dijkstra	1830	deadlock	88	2 (1)	0 (0)	0 (0)	3
			mutual exclusion	1866	0 (0)	3 (1)	0 (0)	
	Berkeley	544	deadlock	15	0 (0)	0 (0)	0 (0)	1
			consistency (1/3)	442	0 (0)	9 (1)	0 (0)	
	Dragon	673	deadlock	19	0 (0)	0 (0)	0 (0)	2
			consistency (7/7)	2015	25 (5)	11 (2)	0 (0)	
	Firefly	469	deadlock	15	0 (0)	0 (0)	0 (0)	1
			consistency (2/4)	617	0 (0)	14 (6)	0 (0)	
	Illinois	490	deadlock	15	0 (0)	0 (0)	0 (0)	1
			consistency (2/2)	184	0 (0)	5 (1)	0 (0)	
	MESI	407	deadlock	14	0 (0)	0 (0)	0 (0)	1
			consistency (2/2)	179	0 (0)	5 (1)	0 (0)	
	MOESI	439	deadlock	14	0 (0)	0 (0)	0 (0)	1
			consistency (7/7)	1496	0 (0)	35 (6)	0 (0)	
	Synapse	398	deadlock	14	0 (0)	0 (0)	0 (0)	1
			consistency (2/2)	18	0 (0)	0 (0)	0 (0)	

9   Conclusion.

We have refined the approach to parameterized verification of systems with regular architectures presented in [15]. Instead of encoding the complete verification question into large, monolithic WS1S-formula, our approach introduces a CE-GAR loop which also outputs an explanation of why the property holds in the form of a typically small set of parameterized invariants (see Example 1). The explanation helps to uncover false positives, where the verification succeeds only because the system or the specification are incorrectly encoded in WS1S. It has also helped to find a subtle bug in the implementation of [15] which hid unnoticed in the complexity of the monolithic formula. Additionally, our incremental approach requires to check smaller WS1S-formulas, which often decreases the verification time (cp. the verification of Dijkstra's mutual exclusion algorithm [15] in 10s to currently 2s).

On the other hand, seeing the abstraction helps one understand the analyzed system. For example, we include in [45] a leader election algorithm for which the parameterization techniques of `ostrich` are too coarse to establish the general safety property of having always at most one leader. However, `ostrich` succeeds to prove the special case that not agents 0 and $n-1$ can become leader at the same time. For this proof `ostrich` finds a family of siphons which hint to a general inductive invariant of the system. Using the semi-automatic mode of

`ostrich` we can then verify this inductive invariant and, as a result of this, the general safety property.

References

1. Abdulla, P.A., Cerans, K., Jonsson, B., Tsay, Y.: General decidability theorems for infinite-state systems. In: LICS. pp. 313–321. IEEE Computer Society (1996)
2. Abdulla, P.A., Delzanno, G., Henda, N.B., Rezine, A.: Regular model checking without transducers (on efficient verification of parameterized systems). In: TACAS. LNCS, vol. 4424, pp. 721–736. Springer (2007)
3. Abdulla, P.A., Jonsson, B., Nilsson, M., Saksena, M.: A survey of regular model checking. In: CONCUR. LNCS, vol. 3170, pp. 35–48. Springer (2004)
4. Abdulla, P.A., Sistla, A.P., Talupur, M.: Model checking parameterized systems. In: Handbook of Model Checking, pp. 685–725. Springer (2018)
5. Apt, K.R., Kozen, D.C.: Limits for automatic verification of finite-state concurrent systems. Information Processing Letters **22**(6), 307 – 309 (1986)
6. Athanasiou, K., Liu, P., Wahl, T.: Unbounded-thread program verification using thread-state equations. In: IJCAR. LNCS, vol. 9706, pp. 516–531. Springer (2016)
7. Außerlechner, S., Jacobs, S., Khalimov, A.: Tight cutoffs for guarded protocols with fairness. In: VMCAI. LNCS, vol. 9583, pp. 476–494. Springer (2016)
8. Baukus, K., Bensalem, S., Lakhnech, Y., Stahl, K.: Abstracting WS1S systems to verify parameterized networks. In: TACAS. LNCS, vol. 1785, pp. 188–203. Springer (2000)
9. Baukus, K., Lakhnech, Y., Stahl, K.: Parameterized verification of a cache coherence protocol: Safety and liveness. In: VMCAI. LNCS, vol. 2294, pp. 317–330. Springer (2002)
10. Bensalem, S., Bozga, M., Nguyen, T., Sifakis, J.: D-Finder: A tool for compositional deadlock detection and verification. In: CAV. LNCS, vol. 5643, pp. 614–619 (2009)
11. Bloem, R., Jacobs, S., Khalimov, A., Konnov, I., Rubin, S., Veith, H., Widder, J.: Decidability of Parameterized Verification. Synthesis Lectures on Distributed Computing Theory, Morgan & Claypool Publishers (2015)
12. Blondin, M., Esparza, J., Helfrich, M., Kucera, A., Meyer, P.J.: Checking qualitative liveness properties of replicated systems with stochastic scheduling. In: CAV (2). LNCS, vol. 12225, pp. 372–397. Springer (2020)
13. Blondin, M., Finkel, A., Haase, C., Haddad, S.: Approaching the coverability problem continuously. In: TACAS. LNCS, vol. 9636, pp. 480–496. Springer (2016)
14. Blondin, M., Finkel, A., Haase, C., Haddad, S.: Approaching the coverability problem continuously. In: TACAS. LNCS, vol. 9636, pp. 480–496. Springer (2016)
15. Bozga, M., Esparza, J., Iosif, R., Sifakis, J., Welzel, C.: Structural invariants for the verification of systems with parameterized architectures. In: TACAS (1). LNCS, vol. 12078, pp. 228–246. Springer (2020)

16. Bozga, M., Iosif, R., Sifakis, J.: Checking deadlock-freedom of parametric component-based systems. In: TACAS (2). LNCS, vol. 11428, pp. 3–20. Springer (2019)
17. Browne, M., Clarke, E., Grumberg, O.: Reasoning about networks with many identical finite state processes. Information and Computation 81(1), $13 - 31$ (1989)
18. Chen, Y., Hong, C., Lin, A.W., Rümmer, P.: Learning to prove safety over parameterised concurrent systems. In: FMCAD. pp. 76–83 (2017)
19. Delzanno, G.: Automatic verification of parameterized cache coherence protocols. In: CAV. pp. 53–68 (2000). https://doi.org/10.1007/10722167_8
20. Desel, J., Esparza, J.: Free choice Petri nets. Cambridge university press (2005)
21. Dijkstra, E.W.: Cooperating Sequential Processes, pp. 65–138. Springer New York, New York, NY (2002). https://doi.org/10.1007/978-1-4757-3472-02
22. Emerson, E.A., Kahlon, V.: Reducing model checking of the many to the few. In: CADE. LNCS, vol. 1831, pp. 236–254. Springer (2000)
23. Emerson, E.A., Namjoshi, K.S.: Reasoning about rings. In: POPL. pp. 85–94 (1995)
24. Esparza, J.: Decidability and complexity of petri net problems - an introduction. In: Petri Nets. Lecture Notes in Computer Science, vol. 1491, pp. 374–428. Springer (1996)
25. Esparza, J.: Parameterized verification of crowds of anonymous processes. In: Dependable Software Systems Engineering, pp. 59–71. IOS Press (2016)
26. Esparza, J., Finkel, A., Mayr, R.: On the verification of broadcast protocols. In: LICS. pp. 352–359. IEEE Computer Society (1999)
27. Esparza, J., Ledesma-Garza, R., Majumdar, R., Meyer, P.J., Niksic, F.: An SMT-based approach to coverability analysis. In: CAV. LNCS, vol. 8559, pp. 603–619. Springer (2014)
28. Esparza, J., Melzer, S.: Verification of safety properties using integer programming: Beyond the state equation. Formal Methods in System Design 16(2), 159–189 (2000)
29. Esparza, J., Meyer, P.J.: An SMT-based approach to fair termination analysis. In: FMCAD. pp. 49–56. IEEE (2015)
30. Esparza, J., Raskin, M., Welzel, C.: Computing parameterized invariants of parameterized petri nets (2021), https://arxiv.org/abs/2103.10280
31. Esparza, J., Raskin, M., Welzel, C.: `ostrich`. https://gitlab.lrz.de/i7/ostrich (2021)
32. Finkel, A., Haddad, S., Khmelnitsky, I.: Minimal coverability tree construction made complete and efficient. In: FoSSaCS. LNCS, vol. 12077, pp. 237–256. Springer (2020)
33. Finkel, A., Schnoebelen, P.: Well-structured transition systems everywhere! Theor. Comput. Sci. 256(1-2), 63–92 (2001)
34. Fribourg, L., Olsén, H.: Reachability sets of parameterized rings as regular languages. Electr. Notes Theor. Comput. Sci. 9, 40 (1997). https://doi.org/10.1016/S1571-0661(05)80427-X
35. Geffroy, T., Leroux, J., Sutre, G.: Occam's razor applied to the petri net coverability problem. Theor. Comput. Sci. 750, 38–52 (2018)
36. German, S.M., Sistla, A.P.: Reasoning about systems with many processes. J. ACM 39(3), 675–735 (1992)
37. Henriksen, J.G., Jensen, J.L., Jørgensen, M.E., Klarlund, N., Paige, R., Rauhe, T., Sandholm, A.: MONA: Monadic second-order logic in practice. In: TACAS. LNCS, vol. 1019, pp. 89–110. Springer (1995)
38. Jacobs, S., Sakr, M.: Analyzing guarded protocols: Better cutoffs, more systems, more expressivity. In: VMCAI. LNCS, vol. 10747, pp. 247–268. Springer (2018)

39. Jensen, H.E., Lynch, N.A.: A proof of burns N-process mutual exclusion algorithm using abstraction. In: TACAS. Lecture Notes in Computer Science, vol. 1384, pp. 409–423. Springer (1998)
40. Kesten, Y., Maler, O., Marcus, M., Pnueli, A., Shahar, E.: Symbolic model checking with rich assertional languages. Theor. Comput. Sci **256**(1), 93 – 112 (2001)
41. Murata, T.: Petri nets: Properties, analysis and applications. Proceedings of the IEEE **77**(4), 541–580 (1989)
42. Reisig, W.: Understanding Petri Nets - Modeling Techniques, Analysis Methods, Case Studies. Springer (2013)
43. Reynier, P., Servais, F.: On the computation of the minimal coverability set of Petri nets. In: RP. LNCS, vol. 11674, pp. 164–177. Springer (2019)
44. The MONA Project: MONA, https://www.bricks.dk/mona
45. Welzel, C., Esparza, J., Raskin, M.: ostrich (Feb 2020). https://doi.org/10.5281/zenodo.4499091
46. Wimmel, H., Wolf, K.: Applying CEGAR to the Petri net state equation. Logical Methods in Computer Science **8**(3) (2012)

A Full description of Example 1

The complete description of the parameterized net $\mathcal{N} = (\mathcal{P}, Tr)$ of the dinning philosophers with one left-handed philosopher is as follows:

- $\mathcal{P} = \{\text{think}, \text{wait}, \text{eat}, \text{free}, \text{taken}\}$.
- $Tr(\boldsymbol{n}, \mathcal{X}, \mathcal{Y}) = \text{GrabFirst} \vee \text{GrabSecond} \vee \text{Release}$, where

$$\text{GrabFirst} := \begin{pmatrix} \exists \boldsymbol{x} . \ 1 \leq \boldsymbol{x} < \boldsymbol{n} \wedge (\mathcal{X}_{\text{think}} = \mathcal{X}_{\text{free}} = \mathcal{Y}_{\text{wait}} = \mathcal{Y}_{\text{taken}} = \{\boldsymbol{x}\}) \\ \wedge (\mathcal{X}_{\text{wait}} = \mathcal{X}_{\text{eat}} = \mathcal{X}_{\text{taken}} = \emptyset) \\ \wedge (\mathcal{Y}_{\text{think}} = \mathcal{Y}_{\text{eat}} = \mathcal{Y}_{\text{free}} = \emptyset) \end{pmatrix}$$
$$\vee$$
$$\begin{pmatrix} (\mathcal{X}_{\text{think}} = \mathcal{Y}_{\text{taken}} = \{0\}) \wedge (\mathcal{X}_{\text{free}} = \mathcal{Y}_{\text{wait}} = \{1\}) \\ \wedge (\mathcal{X}_{\text{wait}} = \mathcal{X}_{\text{eat}} = \mathcal{X}_{\text{taken}} = \emptyset) \\ \wedge (\mathcal{Y}_{\text{think}} = \mathcal{Y}_{\text{eat}} = \mathcal{Y}_{\text{free}} = \emptyset) \end{pmatrix}$$

$$\text{GrabSecond} := \begin{pmatrix} \exists \boldsymbol{x} . \ 1 \leq \boldsymbol{x} < \boldsymbol{n} \ \wedge \ (\mathcal{X}_{\text{wait}} = \mathcal{Y}_{\text{eat}} = \{\boldsymbol{x}\}) \\ \wedge \ (\mathcal{X}_{\text{free}} = \mathcal{Y}_{\text{taken}} = \{\boldsymbol{x} \oplus_{\boldsymbol{n}} 1\}) \\ \wedge \ (\mathcal{X}_{\text{think}} = \mathcal{X}_{\text{eat}} = \mathcal{X}_{\text{taken}} = \emptyset) \\ \wedge \ (\mathcal{Y}_{\text{think}} = \mathcal{Y}_{\text{wait}} = \mathcal{Y}_{\text{free}} = \emptyset) \end{pmatrix}$$
$$\vee$$
$$\begin{pmatrix} (\mathcal{X}_{\text{think}} = \mathcal{Y}_{\text{taken}} = \mathcal{X}_{\text{free}} = \mathcal{Y}_{\text{wait}} = \{0\}) \\ \wedge \ (\mathcal{X}_{\text{wait}} = \mathcal{X}_{\text{eat}} = \mathcal{X}_{\text{taken}} = \emptyset) \\ \wedge \ (\mathcal{Y}_{\text{think}} = \mathcal{Y}_{\text{eat}} = \mathcal{Y}_{\text{free}} = \emptyset) \end{pmatrix}$$

$$\text{Release} := \exists \boldsymbol{x} . \ \boldsymbol{x} < \boldsymbol{n} \ \wedge \ (\mathcal{X}_{\text{eat}} = \mathcal{Y}_{\text{think}} = \{x\} \wedge \mathcal{X}_{\text{taken}} = \mathcal{Y}_{\text{free}} = \{x, x \oplus 1\})$$
$$\wedge \ (\mathcal{X}_{\text{think}} = \mathcal{X}_{\text{wait}} = \mathcal{X}_{\text{free}} = \emptyset)$$
$$\wedge \ (\mathcal{Y}_{\text{wait}} = \mathcal{Y}_{\text{eat}} = \mathcal{Y}_{\text{taken}} = \emptyset)$$

B   Headed rings

Headed rings are very similar to fully symmetric rings but allow for one distinct agent to behave differently. In Example 1 this is the philosopher who takes her forks in an opposite way than all the others. We arbitrarily fix index 0 to be the agent that interacts differently with its neighbors (which are located at indices 1 and $n-1$). Hence, the definition of headed rings coincide mostly with fully symmetric rings but introduces special transitions T_R, and T_L which describe the interaction between indices 0 and 1, and 0 and $n-1$ respectively. Note that most results of the previous section carry over with straightforward adaptions. Moreover, these results are actual generalizations of before since headed rings which identify T_R, T_L and T yield fully symmetric rings.

Definition 7. *A parameterized Petri net $\mathcal{N} = \langle \mathcal{P}, \mathit{Tr} \rangle$ is a* headed ring *if there are finite sets $T_L, T_R, T \subseteq \left(2^{\mathcal{P}} \times 2^{\mathcal{P}} \right) \times \left(2^{\mathcal{P}} \times 2^{\mathcal{P}} \right)$ such that for every $\mathcal{N}(n) = \langle P_n, T_n \rangle$ holds*

$$
T_n = \left(
\begin{aligned}
& \left\{ \left\langle \begin{matrix} R_L \times \{1\} \cup R_R \times \{0\}, \\ Q_L \times \{1\} \cup Q_R \times \{0\} \end{matrix} \right\rangle : \langle\langle R_L, R_R\rangle, \langle Q_L, Q_R\rangle\rangle \in T_L \right\} \\
\cup\, & \left\{ \left\langle \begin{matrix} R_L \times \{0\} \cup R_R \times \{n-1\}, \\ Q_L \times \{0\} \cup Q_R \times \{n-1\} \end{matrix} \right\rangle : \langle\langle R_L, R_R\rangle, \langle Q_L, Q_R\rangle\rangle \in T_R \right\} \\
\cup\, & \left\{ \left\langle \begin{matrix} R_L \times \{i \oplus_n 1\} \cup R_R \times \{i\}, \\ Q_L \times \{i \oplus_n 1\} \cup Q_R \times \{i\} \end{matrix} \right\rangle : \begin{matrix} \langle\langle R_L, R_R\rangle, \langle Q_L, Q_R\rangle\rangle \in T, \\ 0 < i < n-1 \end{matrix} \right\}
\end{aligned}
\right).
$$

The analysis for fully symmetric rings carries over with adaptions that account for the indices $n-1, 0, 1$ having different interactions with each other than the remaining indices.

Proposition 5. *It can be effectively checked if a parameterized Petri net $\langle \mathcal{P}, \mathit{Tr} \rangle$ is a headed ring.*

Once the topology is established, we may use the following results akin to Theorem 1 to generalize local traps:

Theorem 4. *Let $\mathcal{N} = \langle \mathcal{P}, \mathit{Tr} \rangle$ be a headed symmetric ring and let $\langle n, Q \rangle$ be an local indexed trap of $\mathcal{N}$ with a minimal set $\mathbb{I} = \{i_0, \ldots, i_{k-1}\}$ such that $i_j < i_{j+1}$ for all $0 \le j < k-1$ and $Q \cap \mathcal{P} \times [n] \setminus \mathbb{I} = \emptyset$ and $\{0, 1, n-1\} \cap \mathbb{I} = \emptyset$. Then every model of the formula*

$$
\mathit{ParTrap}_{\mathbf{T}}(\mathcal{X}) \coloneqq n \le \boldsymbol{n}
$$

$$
\wedge \exists \boldsymbol{y} : \boldsymbol{y} < \boldsymbol{n} \wedge \left(
\begin{matrix}
\boldsymbol{y} \ne 0, 1, \boldsymbol{n} - 1 \\
\wedge \bigwedge_{0 < j \le k-1} \boldsymbol{y} \oplus_{\boldsymbol{n}} i_j - i_{j-1} \ne 1, 0, \boldsymbol{n} - 1
\end{matrix}
\right)
$$

$$
\wedge \bigwedge_{p \in \mathcal{P}} \forall \boldsymbol{x} : \boldsymbol{x} < \boldsymbol{n} \to \boldsymbol{x} \in \mathcal{X}_p \leftrightarrow \left(
\begin{matrix}
\bigvee_{\langle i_0, p \rangle \in Q} \boldsymbol{x} = \boldsymbol{y} \\
\vee \bigvee_{j > 0, \langle i_j, p \rangle \in Q} \boldsymbol{x} = \boldsymbol{y} \oplus_{\boldsymbol{n}} (i_j - i_{j-1})
\end{matrix}
\right)
$$

$$(8)$$

is an indexed trap of $\mathcal{N}$.

Theorem 5. *Let $\mathcal{N} = \langle \mathcal{P}, \mathit{Tr} \rangle$ be a headed symmetric ring and let $\langle n, Q \rangle$ be an local indexed trap of $\mathcal{N}$ s.t. $Q \cap \mathcal{P} \times [n] \setminus \{0, 1, n-1\} = \emptyset$. Then every model of the formula*

$$ParTrap_{\mathbf{T}}(\mathcal{X}) := n \leq \boldsymbol{n}$$

$$\bigwedge_{p \in \mathcal{P}} \forall \boldsymbol{x} \colon \boldsymbol{x} \in \mathcal{X}_p \leftrightarrow \left(\begin{array}{c} \bigvee_{\langle 0, p \rangle \in Q} \boldsymbol{x} = 0 \\ \bigvee_{\langle 1, p \rangle \in Q} \boldsymbol{x} = 1 \\ \bigvee_{\langle n-1, p \rangle \in Q} \boldsymbol{x} = \boldsymbol{n} - 1 \end{array} \right) \tag{9}$$

is an indexed trap of $\mathcal{N}$.

Adapting Theorem 2 is done similarly by only considering the period of a trap in between the indices 1 and $n - 1$ and formalizing repetitions there while enforcing that indices 0, 1, and $n - 1$ agree with the original trap.