Reachability in Parameterized Systems: All Flavors of Threshold Automata

Jure Kukovec
TU Wien, Favoritenstraße 9–11, 1040 Vienna, Austria
jkukovec@forsyte.at

Igor Konnov
University of Lorraine, CNRS, Inria, LORIA, F-54000 Nancy, France
igor.konnov@inria.fr
https://orcid.org/0000-0001-6629-3377

Josef Widder
TU Wien, Favoritenstraße 9–11, 1040 Vienna, Austria
widder@forsyte.at
https://orcid.org/0000-0003-2795-611X

Abstract

Threshold automata, and the counter systems they define, were introduced as a framework for parameterized model checking of fault-tolerant distributed algorithms. This application domain suggested natural constraints on the automata structure, and a specific form of acceleration, called single-rule acceleration: consecutive occurrences of the same automaton rule are executed as a single transition in the counter system. These accelerated systems have bounded diameter, and can be verified in a complete manner with bounded model checking.

We go beyond the original domain, and investigate extensions of threshold automata: non-linear guards, increments and decrements of shared variables, increments of shared variables within loops, etc., and show that the bounded diameter property holds for several extensions. Finally, we put single-rule acceleration in the scope of flat counter automata: although increments in loops may break the bounded diameter property, the corresponding counter automaton is flattable, and reachability can be verified using more permissive forms of acceleration.

2012 ACM Subject Classification Software and its engineering → Software verification, Theory of computation → Logic and verification, Software and its engineering → Software fault tolerance

Keywords and phrases threshold & counter automata, parameterized verification, reachability

Digital Object Identifier 10.4230/LIPIcs.CONCUR.2018.19

Funding Supported by the Austrian Science Fund (FWF) via the National Research Network RiSE (S11403, S11405), project PRAVDA (P27722), and Doctoral College LogiCS (W1255-N23); and by the Vienna Science and Technology Fund (WWTF) via project APALACHE (ICT15-103).

Acknowledgements We thank the anonymous reviewers for spotting a few technical omissions in the preliminary version of this paper.

1 Introduction

Threshold automata were introduced as a framework for modeling and verification [23, 25, 24, 22] and recently for synthesis [29] of fault-tolerant distributed algorithms. These algorithms typically wait for a quorum of messages, e.g., in replication services, the primary replica may block until it received acknowledgments from a majority of the back-up replicas [28, 33, 14, 34].


© Jure Kukovec, Igor Konnov, and Josef Widder;
licensed under Creative Commons License CC-BY

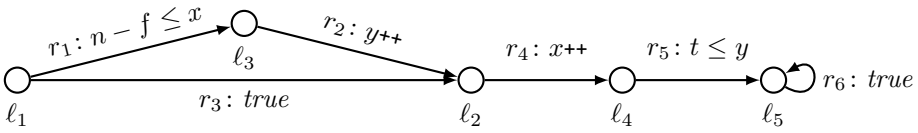

Figure 1 A threshold automaton.

Moreover, these algorithms are parameterized by design, i.e., the number of processes n is a parameter, and consequently, the primary in our example contains a guard that waits for more than $n/2$ messages, a so-called threshold guard. As a result, the local transition relation is parameterized, and therefore these systems are out of reach of classic work in parameterized model checking [15, 7].

We recall the necessary notions of threshold automata by the example in Figure 1. It operates on parameters n, t, and f, and shared variables x and y. The vertices are called locations, and the edges are called rules, which can be guarded, and can increase a shared variable. For instance, the threshold guard in rule r_1 compares the value of variable x to a linear expression over parameters $n - f$. The semantics of threshold automata is defined via counter systems, where a configuration contains the values of shared variables, and a counter value κ_i for each location ℓ_i. The transition relation then is defined by operations on the counters and shared variables. For instance, for some c, if $\kappa_2 \geq c$, then there is a transition defined by rule r_4 that increases κ_4 by c, decreases κ_2 by c and increases x by c.

By allowing arbitrary values of factor c, one obtains a transition relation with a specific form of acceleration (single-rule acceleration), built-in by construction. Then, the transition system is a graph with vertices being configurations, and edges being transitions. By defining paths in this graph, and distances between vertices, one can define the diameter of a transition system. If the diameter d is bounded, then every state is reachable in d steps, and bounded model checking of executions of lengths up to d is a complete verification method for reachability [6]. It was shown [25] that the diameter of transition systems defined by threshold automata is bounded, and in particular, it does not depend on the values of the parameters such as n, t, and f. However, several restrictions on threshold automata were used in [25] to bound the diameter. While these restrictions are well-justified for the original domain of fault-tolerant algorithms, two questions remain open: (i) which of these restrictions were actually necessary to prove the results under single-rule acceleration, and (ii) which restrictions could be avoided by allowing a more permissive form of acceleration?

The purpose of this paper is to explore various extensions of threshold automata, and understand which of them maintain a bounded diameter. We study extensions of the following properties of threshold automata as defined in [25]:

Increments in loops. Canonical threshold automata defined in [25] do not allow updates of shared variables within loops.

Guards. In [25], threshold guards compare shared variables to a threshold, that is, a linear combination of parameters. Since parameter values are fixed in a run, thresholds are effectively constant. As shared variables can only increase in [25], the guards are monotonic; for instance, once the shared variable is greater than the threshold it stays greater, and the evaluation of the guards stays unchanged after that. We consider more general guards: we replace the shared variables (e.g., x) by a function over shared variables, and consider the special case of a difference $(x - y)$, as well as piecewise monotone functions.

Table 1 Summary of results. "p.m. $f(x)$" means a piecewise monotone function of x.

Level	Reversals	Canonical	Bounded diameter?	Flattable?	Decidable reachability?	Class name
x	0	✓	[25, Thm. 8] ✓	✓	✓	TA
p.m. $f(x)$	0	✓	Cor. 18 ✓	✓	✓	PMTA
x	$\leq k$	✓	[27, Thm. 4] ✓	✓	✓	rbTA
x	0	✗	Thm. 9 ✗	Thm. 24 ✓	✓	NCTA
$x - y$	0	✓	✗	✗	Thm. 11 ✗	BDTA
x	∞	✓	✗	✗	Thm. 10 ✗	rTA

Reversibility. In [25], only increments on shared variables are considered because increments are sufficient to model sending a message. As a result, threshold guards were monotone. In this paper, we also consider decrements, which produce schedules that have alternating periods of increasing a variable and decreasing it.

For these extensions, we show that under certain conditions these automata entail bounded diameter results as well. Thus, the diameter result of [25] can be seen as a special case of the results of this paper.

Finally, we consider threshold automata in the scope of counter automata, a modeling framework for infinite-state systems [10, 30, 4]. We consider the concepts of (i) a *flat* counter automaton, whose control graph does not contain nested loops, and (ii) a *flattable* counter automaton, for which a flat counter automaton with the same reachability relation exists. For these automata, there are procedures and tools (FAST) for reachability analysis [30, 4]. We will discuss that the results of [25, 21] imply that canonical threshold automata (no increments in loops) entail flattable counter automata – which explains why FAST verified some benchmarks in the experiments of [25]. Moreover, we show that we can get rid of the canonicity restriction and still prove that the resulting counter automaton is flattable. That is, while non-canonical threshold automata do not fall into the fragment that can be verified with the methods from [25, 21], one can still analyze these automata with more permissive forms of acceleration as implemented in FAST.

An overview of our results is in Table 1, where the simpler classes are at the top; these classes are defined in Section 2.3. The bounded diameter property implies flattability, as we show in Proposition 21, which can be seen in the first three lines. For completeness, in line 3, we mention results on reversal-bounded threshold automata rbTA, which consider the structure of runs rather than threshold automata [27]. Note that flattability of a counter automaton implies that reachability for this automaton is decidable [30].

2 System model

This section generalizes the definitions of [25]. We use the following sets: integers $\mathbb{Z}$ and their extension $\mathbb{Z}_\infty = \mathbb{Z} \cup \{-\infty, +\infty\}$, non-negative integers $\mathbb{N}_0$, reals $\mathbb{R}$. We denote a vector of integers by $\vec{x}$. When the vector dimension is clear, we write $\vec{1}_k$ to denote the unit vector that has 1 at position k and 0 everywhere else, and $\vec{0}$ is the vector filled with zeroes.

2.1 Unrestricted threshold automata

An *unrestricted threshold automaton* (UTA) is a tuple $(\mathcal{L}, \mathcal{I}, \Gamma, \Pi, \mathcal{R})$ where $\mathcal{L}$ is a finite set of *local states* (locations), $\mathcal{I} \subseteq \mathcal{L}$ is a set of *initial local states*, Γ is a finite set of *shared variables*, Π is a finite set of *parameter variables*, and $\mathcal{R}$ is a finite set of *rules*, which are defined below.

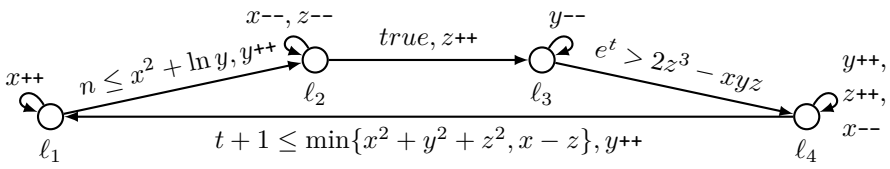

Figure 2 An unrestricted threshold automaton.

Guards. A *nonlinear guard* of a UTA is a formula: $thd(\vec{p}) \bowtie lvl(\vec{x})$, where $\vec{p} = [p_1, \ldots, p_{|\Pi|}]$, $\vec{x} = [x_1, \ldots, x_{|\Gamma|}]$, $lvl\colon \mathbb{Z}^{|\Gamma|} \to \mathbb{R}$ is the *level function*, $thd\colon \mathbb{Z}^{|\Pi|} \to \mathbb{R}$ is the *threshold function*, and $\bowtie$ is one of $\{<, \leq, >, \geq\}$. When $\bowtie$ is either $<$ or $\leq$, the guard is called a *lower guard*, otherwise it is an *upper guard*. For $x \in \Gamma$ and $a_0, a_1, \ldots, a_{|\Pi|} \in \mathbb{Z}$, a guard of the following form is called *affine*: $a_0 + \sum_{i=1}^{|\Pi|} a_i p_i \bowtie x$. (Affine guards have only one shared variable.)

Rules. A *rule* is a tuple $(from, to, \Phi, \vec{u})$ where $to, from \in \mathcal{L}$ are two local states, Φ is a set of nonlinear guards and $\vec{u} \in \mathbb{Z}^{|\Gamma|}$ is an update vector.

▶ **Example 1.** Consider the automaton in Figure 2, demonstrating the nonlinear guards and rules that are *not* considered in [25]. ◁

2.2 Semantics of UTA: counter systems

Configurations. For a UTA $A = (\mathcal{L}, \mathcal{I}, \Gamma, \Pi, \mathcal{R})$, a triple of vectors $(\vec{\kappa}, \vec{g}, \vec{p}) \in \mathbb{N}_0^{|\mathcal{L}|} \times \mathbb{Z}^{|\Gamma|} \times \mathbb{N}_0^{|\Pi|}$ is called a *configuration*. The vectors have the following meaning: vector $\vec{\kappa} \in \mathbb{N}_0^{|\mathcal{L}|}$ stores the values of the location counters, vector $\vec{g} \in \mathbb{Z}^{|\Gamma|}$ stores the values of the shared variables, and vector $\mathbb{N}_0^{|\Pi|}$ stores the parameters.

Transitions. Given a UTA $A = (\mathcal{L}, \mathcal{I}, \Gamma, \Pi, \mathcal{R})$, a *transition* is a pair $(rule, factor)$ where $rule \in \mathcal{R}$ and $factor \in \mathbb{N}_0$. Note that the single-rule acceleration is built into to the definition of a transition, by allowing $factor > 1$. We use the notation $t.rule$ and $t.factor$ to refer to the tuple elements of the same name. Additionally, for any tuple field e of $t.rule$ we shorten $t.rule.e$ to $t.e$ for brevity (e.g., $t.rule.from$ becomes $t.from$).

Given a configuration σ and a formula φ over the shared variables Γ and parameters Π, we will use the notation $(\sigma.\vec{g}, \sigma.\vec{p}) \models \varphi$, or just $\sigma \models \varphi$, to mean that the formula φ holds true when the shared variables and the parameters are substituted with their respective values from $\sigma.\vec{g}$ and $\sigma.\vec{p}$.

We say that a *rule* r *is unlocked* in a configuration σ if $(\sigma.\vec{g}, \sigma.\vec{p}) \models \bigwedge_{\varphi \in r.\Phi} \varphi$. Further, a transition $t = (r, a)$ is unlocked in a configuration σ if r remains unlocked after at least $a - 1$ updates imposed by $r.\vec{u}$, that is, for each $k \in \{0, 1, \ldots, a-1\}$, it holds that $(\sigma.\vec{g} + k \cdot r.\vec{u}, \sigma.\vec{p}) \models \bigwedge_{\varphi \in r.\Phi} \varphi$.

▶ **Definition 2.** A transition $t = (r, a)$ is *applicable to* a configuration σ if t is unlocked in σ and $\sigma.\vec{\kappa}[r.from] \geq a$. When t is applicable to σ, we call σ' the result of applying t to σ – denoted as $t(\sigma)$ – if the requirements 1–3 are met:

1. the location counters are changed by a, that is, $\sigma'.\vec{\kappa} = \sigma.\vec{\kappa} + a \cdot (\vec{1}_{r.to} - \vec{1}_{r.from})$,
2. the update vector is added a times to the shared variables: $\sigma'.\vec{g} = \sigma.\vec{g} + a \cdot r.\vec{u}$,
3. the parameters do not change: $\sigma'.\vec{p} = \sigma.\vec{p}$.

Definition 2 explicitly allows successive applications of the same rule to be compressed into a single transition. This kind of acceleration was introduced in [25], and we call it *single-rule* acceleration.

▶ **Example 3.** Consider the automaton in Figure 1. The following table shows a configuration σ_0, and configurations σ_1 and σ_2 after applying one and two transitions, respectively, to the configuration σ_0:

configuration	counters $\vec{\kappa}$	shared variables $\vec{g}$	parameters $\vec{p}$
σ_0	$(\mathbf{4}, 0, 0, 0, 0)$	$(0, 0)$	$(4, 1, 1)$
σ_1	$(\mathbf{2}, \mathbf{2}, 0, 0, 0)$	$(0, 0)$	$(4, 1, 1)$
σ_2	$(2, \mathbf{1}, 0, \mathbf{1}, 0)$	$(\mathbf{1}, 0)$	$(4, 1, 1)$

First, the parameters are initialized to $n = 4, t = f = 1$, and the counter of location ℓ_1 equals to n (configuration σ_0). Then, transition $(r_3, 2)$ is applied to σ_0, resulting in the counter of ℓ_2 increasing by 2 and the counter of ℓ_1 decreasing by 2 in configuration σ_1. Finally, rule r_4 is executed once, incrementing x to obtain σ_2. ◁

Number of instances. As in [25], we assume that a threshold automaton is equipped with a function $N : \mathbb{N}_0^{|\Pi|} \to \mathbb{N}_0$. Intuitively, every configuration σ captures a state of $N(\sigma.\vec{p})$ instances of the threshold automaton. The authors of [25] did not restrict function N, as they were concerned only with the length of the shortest sequences of transitions connecting any two configurations. In this paper, we assume that the relation $\{(\vec{p}, N(\vec{p})) \colon \vec{p} \in \mathbb{N}_0^{|\Pi|}\}$ can be defined with a formula in Presburger arithmetic. In Example 3, we define N with the following formula over the parameters n, t, and f as well as the outcome of the function N: $(n > 3t \to N = n - f \wedge f \geq 0 \wedge t \geq 0) \wedge (n \leq 3t \to N = 0)$.

In our example, the number N is positive only if $n > 3t$, and equals to zero otherwise. This allows us to prune "irrelevant" parameter values. (In distributed computing, this is achieved by writing a so-called *resilience condition*.)

Counter systems. Having defined the configurations and transitions, we define a counter system of a threshold automaton:

▶ **Definition 4.** Given a UTA $A = (\mathcal{L}, \mathcal{I}, \Gamma, \Pi, R)$, we define its *counter system* $CS(A)$ as a transition system (Σ, I, R), where:

- Σ is the set of all possible *configurations*.
- $I \subseteq \Sigma$ is the set of *initial configurations*; their counter values in the initial locations sum up to $N(\vec{p})$. Formally, a configuration $\sigma_0 \in \Sigma$ belongs to I if and only if the following conditions hold: $\sigma_0.\vec{\kappa}[\ell] = 0$ for $\ell \in \mathcal{L} \setminus \mathcal{I}$ and $N(\sigma_0.\vec{p}) = \sum_{\ell \in \mathcal{I}} \sigma_0.\vec{\kappa}[\ell]$, as well as, $\sigma_0.\vec{g} = \vec{0}$.
- $R \subseteq \Sigma \times \Sigma$ is the *transition relation*. A pair of configurations (σ, σ') belongs to R if and only if there is a transition t that is applicable to σ, and $\sigma' = t(\sigma)$.

A *schedule* is a finite sequence of transitions. A schedule $\tau = t_1, \ldots, t_m$ is *applicable* to a configuration σ_0 if there exists a sequence of configurations $\sigma_1, \ldots, \sigma_m$ where $\sigma_i = t_i(\sigma_{i-1})$ for all $0 < i \leq m$. We define $\tau(\sigma_0)$ to be σ_m. We denote the concatenation of schedules τ and τ' by $\tau \cdot \tau'$ and the length of a schedule $\tau = t_1, \ldots, t_m$ as $|\tau| = m$. By ϵ, we refer to the empty schedule, which has length 0 and satisfies $\epsilon(\sigma) = \sigma$ for all σ in Σ.

For a schedule $\tau = t_1, \ldots, t_n$ and two indices $i, j \in \mathbb{Z}$, we define the subschedule $\tau_{[i,j]}$ as follows ($\tau_{[i,j)}$, $\tau_{(i,j]}$, and $\tau_{(i,j)}$ are obtained by choosing the intervals accordingly):

$$\tau_{[i,j]} = \begin{cases} t_{\max(1,i)}, \ldots, t_{\min(n,j)}, & \text{when } i \leq j, \\ \epsilon, & \text{when } i > j \end{cases}$$

We say that a configuration σ' is *reachable* from a configuration σ, if there is a schedule τ with the following properties: (1) τ is applicable to σ, and (2) $\tau(\sigma) = \sigma'$.

Bounded diameters. The central result of [25] is that for counter systems of threshold automata one can check, whether one configuration is reachable from another. It is sufficient to inspect the schedules of length within a precomputed bound on the diameter:

▶ **Definition 5.** Given a UTA A and its counter system $CS(A) = (\Sigma, I, R)$, a number $d \in \mathbb{N}_0 \cup \{\infty\}$ is the *diameter* of $CS(A)$ if d is the smallest number with the following property:

For every pair of configurations $\sigma, \sigma' \in \Sigma$, if σ' is reachable from σ, then there is a schedule τ such that: (a) τ is applicable to σ, (b) $\tau'(\sigma) = \sigma'$, and (c) $|\tau'| \leq d$.

One of our contributions is in finding fragments of unrestricted threshold automata whose counter systems have a bounded diameter. In Section 4, we give examples of UTA whose counter systems have unbounded diameters. Moreover, we show that there are classes of UTA, for which the following problem – which generalizes the problem from [21] – is undecidable:

Parameterized reachability. Given a UTA $A = (\mathcal{L}, \mathcal{I}, \Gamma, \Pi, R)$, a *state property* B is a Boolean combination of formulas that have the form $\vec{\kappa}[\ell] = 0$, for some $\ell \in \mathcal{L}$. The *parameterized reachability* problem is to decide whether there are parameter values $\vec{p} \in \mathbb{N}_0^{|\Pi|}$, an initial configuration $\sigma_0 \in I$, with $\sigma_0.\vec{p} = \vec{p}$, and a schedule τ, such that τ is applicable to σ_0, and property B holds in the final state: $\tau(\sigma_0) \models B$.

2.3 Fragments of unrestricted threshold automata

In order to prove the bounded diameter property, we consider various restrictions on the guards, updates, the transition relation, and other aspects of UTA. The first restriction prohibits modifications of shared variables in loops [25]:

▶ **Definition 6.** A rule r lies on a cycle, if there is a sequence of rules $r_0, \ldots, r_k$, where $r = r_0$ and $r_i.to = r_j.from$ for $0 \leq i \leq k$ and $j = i + 1 \mod (k+1)$.
A UTA is *canonical* if $r.\vec{u} = \vec{0}$ for every rule $r \in \mathcal{R}$ that lies on a cycle.

Canonical Threshold Automata (TA). This class contains UTAs with the following properties: (1) they are canonical, (2) all guards are affine, and (3) the update vectors in all rules are non-negative. This is the class of automata considered in [25, 24], which is known to have a bounded diameter:

▶ **Theorem 7** ([25]). *For every TA A, there exists a constant $\mathcal{C}$, such that the diameter of the associated counter system is less than or equal to $d(CS(A)) = (\mathcal{C}+1) \cdot |\mathcal{R}| + \mathcal{C}$ (independently of the parameters).*

Piecewise Monotone Threshold Automata (PMTA). This class contains UTAs with the following properties: (1) they are canonical, (2) all level functions in the guards are piecewise monotone[1], and (3) the update vectors in all rules are non-negative.

Bounded Difference Threshold Automata (BDTA). This class contains UTAs with the following properties: (1) they are canonical, (2) all level functions in the guards are of the form x_i or $x_i - x_j$ for some $x_i, x_j \in \Gamma$, and (3) the update vectors in all rules are non-negative.

[1] The domain of a piecewise monotone function can be decomposed into finitely many intervals where the function is monotone.

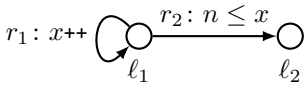

Figure 3 A simple NCTA.

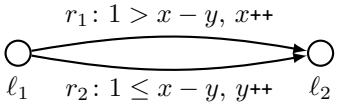

Figure 4 A BDTA with unbounded diameter.

Non-canonical generalizations of TA, PMTA, and BDTA. For the mentioned classes, we omit the requirement of the automaton being canonical, and denote these classes as: NCTA, NCPMTA, and NCBDTA.

Reversible of TA, PMTA, and BDTA. For the mentioned classes, we allow shared variables to be both increased and decreased, and denote these classes as: rTA, rPMTA, and rBDTA.

Reversal-bounded extensions of TA, PMTA, and BDTA. To introduce reversal-bounded automata, we need the following definition.

▶ **Definition 8.** A schedule $t_1 \cdot \tau \cdot t_2$ is an x-reversal if: (a) one of the transitions t_1 or t_2 increases x and the other decreases x, that is, $t_1.\vec{u}[x] \cdot t_2.\vec{u}[x] < 0$, and (b) every transition t in τ does not update x, that is, $t.\vec{u}[x] = 0$. If for every shared variable x, the number of x-reversals in a schedule is at most N, the schedule is called N-*reversible*.

Similar to reversal-bounded counter machines [20], we define the classes rbTA, rbPMTA, and rbBDTA by restricting the counter systems of the respective reversible automata to N-reversible schedules (where N is fixed).

3 Negative results: unbounded diameters and undecidability

We give examples of NCTA and BDTA whose counter systems have unbounded diameters. Then, we show that reachability is undecidable for counter systems of BDTA and rTA.

3.1 Unbounded diameters of non-canonical threshold automata

When we permit shared variables to be updated within loops, the diameter of the counter system becomes unbounded:

▶ **Theorem 9.** *There is an NCTA whose counter system has unbounded diameter.*

Proof. Figure 3 shows such an NCTA, where x is the only shared variable, and n the only parameter. To prove the theorem, take the configuration σ with $\sigma.\vec{\kappa} = (1, 0)$, $\sigma.\vec{g} = (0)$, and $\sigma.\vec{p} = (n)$ for $n > 0$. We show that the following configuration σ' can be reached from σ in no less than $n + 1$ transitions: $\sigma'.\vec{\kappa} = (0, 1)$, $\sigma'.\vec{g} = (n)$, and $\sigma'.\vec{p} = (n)$. In σ, rule r_2 is locked, and rule r_1 is not, so r_1 must be used at least n times to unlock r_2. Since the sum of the values of location counters initially is 1 and is invariant, we can only use transitions with a factor of at most 1. Thus, to reach σ' from σ, we have to execute n copies of the transition $(r_1, 1)$ and then the transition $(r_2, 1)$. Hence, the diameter must be at least $n + 1$, and thus grows with the unbounded parameter n. ◀

The automaton in Figure 3 encodes the simple loop "`while (n <= x) x++;`" One can argue that this automaton can be accelerated by compressing self-loops into one transition; which requires another form of acceleration. Figure 5 shows an example that cannot be easily fixed by this. This example can be treated with more general acceleration techniques, as demonstrated in Section 6.

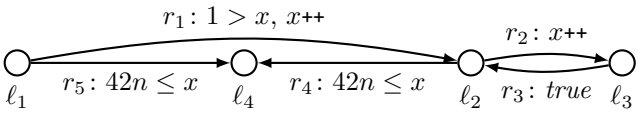

Figure 5 A non-canonical automaton with unbounded diameter.

3.2 Undecidability for reversible and bounded-difference automata

We show even stronger results for rTA and BDTA: reachability is undecidable and thus counter systems of such automata cannot be analyzed with any form of acceleration.

▶ **Theorem 10.** *Parameterized reachability for counter systems of loop-free rTA is undecidable.*

Proof. We use rTA to encode two-counter machines 2CM, for which the halting problem is undecidable [32]. A command of a 2CM is a triple (`from`, `cmd`, `to`) where `from` and `to` are labels from the set $\{1, \ldots, m\}$ for some m, and `cmd` is one of the operations: `inc` x, `dec` x, `inc` y, `dec` y, `zero?` x, `zero?` y. The label m designates the halting command. For the two counters we use two shared variables x and y. For each label i we also add a shared variable at_i, that we use as a Boolean flag to indicate whether the 2CM currently is at label i. There is also a shared variable *init*, which is used for initialization.

It remains to encode the control structure of a 2CM (which may contain loops) in a threshold automaton without loops. Our rTA has three locations ℓ_0, ℓ_1, ℓ_2, where ℓ_0 is the initial one. First, we introduce a special initialization rule from ℓ_0 to ℓ_1 that is guarded with $init < 1$ and increments *init*. Second, for each command we introduce a rule from ℓ_0 to ℓ_1. For command (i, cmd, j), the rule is guarded with $at_i > 0 \land init \geq 1 \land init < 2$, and e.g., $0 \geq x \land 0 \leq x$, if the test for zero is needed. The update of the rule contains at_i`--` and at_j`++` (goto label j from label i), and the required increment/decrement of a counter as e.g., x`++` or y`--`. Third, the last rule detects that the 2CM halted: it goes from ℓ_0 to ℓ_2 and is guarded with $at_m \geq 1$.

The number of instances is $N(n) = n + 2$ for the only parameter n. Thus, n steps of the 2CM are modeled by $n + 1$ transitions of the constructed counter system; the $(n + 2)$th transition may move at least one automaton to the location ℓ_2. Hence, the counter system simulates arbitrarily many steps of the 2CM. We ask the parameterized reachability question of whether the counter system reaches a configuration σ with $\sigma.\vec{\kappa}[\ell_2] \neq 0$ (for some value of n). A positive answer is given *if and only if* the 2CM halts; undecidability follows. ◀

Now we consider BDTA. Figure 4 shows an example of a BDTA whose counter system has unbounded diameter: Every schedule allowed by this threshold automaton is an alternating sequence of the transitions $(r_1, 1)$ and $(r_2, 1)$. Thus to increase the counter κ_2 to n, we require a schedule of length n, which is an unbounded parameter. This shows that sinlge-rule acceleration does not help us to analyze BDTA. In fact, no form of acceleration helps:

▶ **Theorem 11.** *Parameterized reachability for counter systems of loop-free BDTA is undecidable.*

The proof goes along the same lines as the proof of Theorem 10. The only complication is to encode a decrement of a shared variable by using increments and bounded differences. To this end, for each variable x, we introduce two shared variables x_1 and x_2. The difference $x_1 - x_2$ simulates a counter x. Whenever x has to be decremented, we increment x_2, and when x has to be incremented, we increment x_1. A test $x = 0$ is simulated as a conjunction $0 \geq x_1 - x_2 \land 0 \leq x_1 - x_2$.

4 Positive results: bounding the diameter

We extend the framework and the proofs of [25] to prove the bounded diameter property for certain fragments of UTAs. A key observation in [25] is that if shared variables are only increased, then the evaluation of every (affine) threshold guard changes at most once in a schedule. This argument obviously applies even if increments occur in loops:

▶ **Proposition 12** (Monotonicity of affine guards). *For an NCTA configuration σ, if a transition t is applicable to σ, then the following holds:*

1. *For a lower affine guard φ:*
   (a) If $\sigma \models \varphi$, then $t(\sigma) \models \varphi$, and (b) if $t(\sigma) \not\models \varphi$, then $\sigma \not\models \varphi$.
2. *For an upper affine guard φ:*
   (c) If $\sigma \not\models \varphi$ then $t(\sigma) \not\models \varphi$, and (d) if $t(\sigma) \models \varphi$, then $\sigma \models \varphi$.

4.1 A sufficient condition for diameter boundedness

Proposition 12 does not apply to unrestricted threshold automata for two reasons: First, NCTA only allow shared variables to be incremented, whereas UTA allow both increments and decrements. Obviously, an affine threshold guard such as $n \leq x$ can change its evaluation arbitrary many times, if increments and decrements of x are alternated (as parameter n is constant in a schedule). Second, even if we restrict updates of shared variables to non-negative vectors, guards such as $0 \leq x - y$ can change their evaluations arbitrarily often in a single schedule (cf. Theorem 11).

Proposition 12 implies that for every (affine) guard φ, when a schedule τ is applied to a configuration σ, schedule τ can be split into two intervals: $\tau_{[1,k)}$ and $\tau_{[k,|\tau|]}$ with the following property: $\tau_{[1,i)}(\sigma) \models \varphi$ iff $\sigma \models \varphi$ for $1 \leq i \leq k$, and $\tau_{[1,j]}(\sigma) \not\models \varphi$ iff $\tau(\sigma) \not\models \varphi$ for $k \leq j \leq |\tau|$. In other words, the evaluation of φ may only change in the transition from $\tau_{[1,k-1]}(\sigma)$ to $\tau_{[1,k]}(\sigma)$. We extend this idea to non-linear guards by requiring the guards to preserve their evaluations in a bounded number of intervals. In face of Theorem 11, we thus impose two restrictions on UTA: (1) we allow only non-negative updates of shared variables, and (2) we allow level functions to change evaluation of the guards a bounded number of times.

Consider a guard φ, a configuration σ, and a schedule τ applicable to σ. We say that τ is *steady* with respect to (φ, σ), if it has the following property: $\tau_{[1,i]}(\sigma) \models \varphi$ if and only if $\sigma \models \varphi$ for $1 \leq i \leq |\tau|$.

▶ **Definition 13** (Bounded steadiness). We say that a guard φ of a UTA A is *bounded-steady* w.r.t A, if there exists a number $N \geq 0$, called the *flip bound* of φ, with the following property:

For every configuration σ of the counter system of A and every schedule $\tau = t_1, \ldots, t_n$ applicable to σ, there is a sequence of indices $0 = i_0 \leq i_1 \leq \cdots \leq i_N \leq i_{N+1} = n + 1$ such that $\tau_{(i_j, i_{j+1})}$ is steady with respect to φ and $\tau_{[1,i_j]}(\sigma)$ for $0 \leq j \leq N$.

Bounded-steadiness is central in proving the bounded diameter property:

▶ **Theorem 14** (Bounded diameter criterion). *Every canonical UTA A with non-negative updates of shared variables satisfies the following:*

If every guard is bounded-steady w.r.t. A, then the diameter of the counter system $CS(A)$ is bounded by a constant.

In the context of TA, constructions are introduced in [25] to remove cycles and reorder transitions (to apply acceleration), in order to shorten subschedules in which evaluations of guards do not change, i.e., steady subschedules. The results of [25] can be summarized in the following lemma.

▶ **Lemma 15.** *There exists an total order of rules $\prec$ such that for every schedule τ, there exists a unique schedule,* $\mathrm{short}(\tau)$*, with the following properties:*

1. *If transition (r, a) appears in* $\mathrm{short}(\tau)$*, then τ contains a transition (r, a') for some a'.*
2. *If transition (r, a) appears before (r', a') in* $\mathrm{short}(\tau)$*, then $r \prec r'$.*
3. *If for a configuration σ, an applicable schedule τ is steady with respect to all guards and σ, then* $\mathrm{short}(\tau)$ *is applicable to σ and $\tau(\sigma) = \mathrm{short}(\tau)(\sigma)$.*

One can prove the above lemma independently of the shape of the guards. For the proof one only uses that in a steady schedule the evaluation of guards does not change. As a result, one can directly apply the proofs from [25] to generalize Lemma 15 to UTA. This allows us to replace a steady schedule τ by $\mathrm{short}(\tau)$, which reaches the same state and whose length is bounded by Lemma 15(2), because threshold automata have a fixed number of rules and $\prec$ is a total order. What remains to be proven for Theorem 14 is that every schedule of a threshold automaton with bounded-steady guards can be decomposed into a bounded number of steady subschedules.

Proof of Theorem 14. Let $\varphi_1, \ldots, \varphi_m$ be the bounded-steady guards. Let σ be a configuration and $\tau = t_1, \ldots, t_n$ a schedule applicable to it. Since each φ_j is bounded-steady it has a flip bound N_j, for which there exist $i^j_1, \ldots, i^j_{N_j}$ with the property that $\tau_{(i^j_k, i^j_{k+1})}$ is steady with respect to φ_j and $\tau_{[1,i^j_k]}(\sigma)$ for $0 \leq k \leq N_j$. We denote by S^i_j the set of critical indices $\{i^j_1, \ldots, i^j_{N_j}\}$.

We denote by S the set $\bigcup_{j=1}^m S_j$, and by $i_1, \ldots, i_l$ its elements. Additionally, denote $i_0 = 0$ and $i_{l+1} = n + 1$. The set S partitions τ into finer subschedules than each S_j, that is, for every $0 \leq k \leq l$ and for every $1 \leq j \leq m$ there is an index $i^j_p \in S_j$ such that the schedule $\tau_{(i_k, i_{k+1})}$ is a subschedule of the steady schedule $\tau_{(i^j_p, i^j_{p+1})}$. Because a subschedule of a steady schedule is also steady by definition (w.r.t. its initial configuration and the same guard), we can conclude that the schedules $\tau_{(i_k, i_{k+1})}$ are steady with respect to all guards φ_j and $\tau_{[1,i_k]}(\sigma)$.

We can therefore apply Lemma 15 to each $\tau_{(i_k, i_{k+1})}$ and replace it with a shortened schedule. By property (2) of Lemma 15 and because $\prec$ is a total order, the length of the shortened schedules is at most $|\mathcal{R}|$. After replacing every $\tau_{(i_k, i_{k+1})}$ with $\mathrm{short}(\tau_{(i_k, i_{k+1})})$, we obtain a schedule τ', which is applicable to σ, has the property that $\tau'(\sigma) = \tau(\sigma)$ and $|\tau'| \leq (|S|+1) \cdot |\mathcal{R}| + |S|$. By the definition of S, it holds that $|S| \leq \sum_{j=1}^m |S_j| \leq \sum_{j=1}^m N_j$. ◀

4.2 Two fragments with bounded-steady guards

Theorem 14 gives us a sufficient condition for a function to be used in a guard so that the resulting counter system has a bounded diameter. The condition applies to threshold automata with non-negative updates to shared variables. Thus, we can characterize bounded-steady guards by the shape of their level functions.

▶ **Proposition 16.** *Every canonical UTA A with non-negative updates of shared variables has the following property: If a threshold guard φ has the shape $thd(\vec{p}) \bowtie F(y)$ for a shared variable $y \in \Gamma$, a comparison $\bowtie \in \{<, \leq, >, \geq\}$, and a piecewise-monotone function $F \colon \mathbb{Z} \to \mathbb{R}$, then the guard φ is bounded-steady w.r.t. A.*

▶ **Example 17.** Consider piecewise-monotone functions $f_1(x)$, $f_2(x)$ and reals $a_n, \ldots a_0, b \in \mathbb{R}$ with $b > 0$. Then, $a_n \cdot x^n + \cdots + a_1 \cdot x + a_0$, b^x, $\ln x$, and $\min\{f_1(x), f_2(x)\}$ are piecewise-monotone functions of $x \in \mathbb{Z}$. Each of them can be used as $F(x)$ in Proposition 16, and thus they produce bounded-steady guards. ◁

As a corollary of Proposition 16 and Theorem 14, the threshold automata with piecewise-monotone functions in the guards have the bounded diameter property:

▶ **Corollary 18.** *For every PMTA, the diameter of its counter system is bounded.*

Note that the affine threshold guards of [25] have the shape required in Proposition 16, and thus are just a special case.

We generalize Proposition 16 to guards over multiple shared variables. Recall that an m-dimensional integer box is a product of m intervals, that is, $B = \mathbb{Z}^m \cap [a_1, b_1] \times \cdots \times [a_m, b_m]$ for some boundaries $a_1, b_1, \ldots, a_m, b_m \in \mathbb{Z}_\infty$.

▶ **Proposition 19.** *Consider a UTA with non-negative updates of shared variables. A non-linear guard $thd(\vec{p}) \bowtie lvl(\vec{x})$, for $\bowtie \in \{<, \leq, >, \geq\}$, is bounded-steady, if:*
For every level $C \in \mathbb{R}$, the function domain $\mathbb{Z}^{|\Gamma|}$ of the level function can be partitioned into a finite set of disjoint $|\Gamma|$-dimensional boxes $B_1, \ldots, B_k$ that satisfy $\{\vec{x} \in B_i \mid C \bowtie lvl(\vec{x})\}$ is equal to either B_i or $\emptyset$ for $1 \leq i \leq k$.

As a result, the following two-variable functions give us bounded-steady guards:

$$x + y, \quad x \cdot y, \quad \min(f_1(x), f_2(y)) \text{ or } \max(f_1(x), f_2(y)) \text{ for piecewise-monotone } f_1 \text{ and } f_2$$

5 Relation to flattable counter automata

Counter automata model infinite-state systems and have acceleration procedures and tools for reachability analysis [10, 30, 4]. Threshold automata give rise to accelerated counter systems. In this section, we establish a link between these two frameworks. In particular, from a threshold automaton A, we construct two kinds of counter automata: $\mathsf{CA}^0(A)$ is a counter automaton that executes a single UTA rule without any built-in acceleration, and $\mathsf{CA}^1(A)$ is a counter automaton that executes one UTA rule several times in one step. The automaton $\mathsf{CA}^1(A)$ corresponds to our counter system $CS(A)$ in Section 2.2. In our analysis, single-rule acceleration plays a central role in finding diameter bounds, whereas the procedures for counter automata employ more general forms of acceleration. In fact, $\mathsf{CA}^0(A)$ and $\mathsf{CA}^1(A)$ have the same reachability relation, and any of them can in principle be used as the input to the techniques for counter automata.

We recall the definitions of counter automata from [30], operating on m counters.

▶ **Definition 20.** An m-dimensional counter automaton CA is defined as a tuple $(Q, T, \mathsf{src}, \mathsf{tgt}, \{G_t\}_{t \in T})$ with the following properties:

- Q and T are finite, non-empty sets of CA-*locations* and CA-*transitions* respectively,
- $\mathsf{src} : T \to Q$ and $\mathsf{tgt} : T \to Q$ are the *source* and *target* mappings respectively, and
- $\{G_t\}_{t \in T}$ is a finite family of binary relations on $\mathbb{N}^m$ called *flow guards*.

The semantics of the counter automaton CA is defined as a transition system $(\mathcal{C}_{\mathsf{CA}}, \to_{\mathsf{CA}})$ with the following properties:

1. The set $\mathcal{C}_{\mathsf{CA}} = Q \times \mathbb{N}_0^m$ captures CA-configurations, and
2. the relation $\to_{\mathsf{CA}} \subseteq \mathcal{C}_{\mathsf{CA}} \times \mathcal{C}_{\mathsf{CA}}$ captures CA-steps. CA makes a step from a configuration $(q, \vec{x}) \in \mathcal{C}_{\mathsf{CA}}$ to a configuration $(q', \vec{x}') \in \mathcal{C}_{\mathsf{CA}}$ via a transition $t \in T$ – formally written as $(q, \vec{x}) \to_{\mathsf{CA}} (q', \vec{x}')$ – if the following holds:

$$q = \mathsf{src}(t) \text{ and } q' = \mathsf{tgt}(t) \text{ and } (\vec{x}, \vec{x}') \in G_t$$

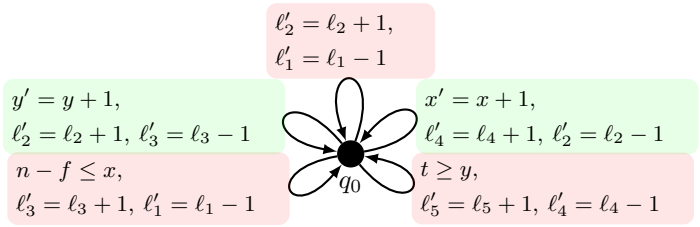

Figure 6 A counter automaton for the threshold automaton in Figure 1.

A sequence $(q_1, \vec{x}_1), \ldots, (q_k, \vec{x}_k)$ of CA-configurations is called a CA-*path*, if $(q_i, \vec{x}_i) \to_{\mathsf{CA}} (q_{i+1}, \vec{x}_{i+1})$ for $1 \leq i < k$. Then, the *reachability relation* $\to^*_{\mathsf{CA}} \subseteq \mathbb{N}_0^{|m|} \times \mathbb{N}_0^{|m|}$ contains all the pairs of vectors that are connected with a path for some control locations, that is, $\vec{x} \to^*_{\mathsf{CA}} \vec{x}'$ if and only if there is a CA-path $(q_1, \vec{x}_1), \ldots, (q_k, \vec{x}_k)$ with $\vec{x} = \vec{x}_1$ and $\vec{x}' = \vec{x}_k$.

A counter automaton without acceleration. Fix an unrestricted threshold automaton $A = (\mathcal{L}, \mathcal{I}, \Gamma, \Pi, \mathcal{R})$, and let P be the set of variables $\mathcal{L} \cup \Gamma \cup \Pi$. To represent the configurations of the UTA counter system, we use vectors $\vec{x} = (x_1, \ldots, x_{|P|}) \in \mathbb{N}_0^{|P|}$, where each element x_i stores the value of a variable from the set P (there is a bijection). For a vector $\vec{x} \in \mathbb{N}_0^{|P|}$ and a set $U \subseteq P$, with $x|_U$, we denote the projection of $\vec{x}$ on the variables from U.

A $|P|$-dimensional counter automaton $\mathsf{CA}^0(A) = (Q, T, \mathsf{src}, \mathsf{tgt}, \{G_t\}_{t \in T})$ is constructed as follows:

- The automaton has only one CA-location, that is, $Q = \{q_0\}$ for some q_0,
- The CA-transitions are identical to the UTA rules, that is, $T = \mathcal{R}$,
- Every transition $t \in T$ originates from the location q_0 and ends in q_0; formally, $\mathsf{src}(t) = \mathsf{tgt}(t) = q_0$,
- For every rule $r \in \mathcal{R}$, the flow relation $G_r \subseteq \mathbb{N}_0^{|P|} \times \mathbb{N}_0^{|P|}$ is the intersection of two relations Guard_r and Update_r that are defined as:

$$(\vec{x}, \vec{x}') \in \mathsf{Guard}_r \text{ if and only if } (\vec{x}|_\Gamma, \vec{x}|_\Pi) \models \bigwedge_{\varphi \in r.\Phi} \varphi$$

$$(\vec{x}, \vec{x}') \in \mathsf{Update}_r \text{ if and only if } \vec{x}'|_\Pi = \vec{x}|_\Pi, \tag{1}$$

$$\vec{x}'|_\Gamma = \vec{x}|_\Gamma + r.\vec{u}, \text{ and } \vec{x}'|_\mathcal{L} = \vec{x}|_\mathcal{L} + \vec{1}_{r.to} - \vec{1}_{r.from} \tag{2}$$

Given the threshold automaton in Figure 1, we construct the respective counter automaton in Figure 6. Apart from the shared variables and parameters, the counter automaton explicitly maintains a counter for each location of UTA, whereas in threshold automata these counters are implicit.

A counter automaton with single-rule acceleration. Given a UTA A, we define its counter automaton with single-rule acceleration $\mathsf{CA}^1(A)$. This automaton is structurally the same as $\mathsf{CA}^0(A)$, except that the flow relation G_r for $r \in \mathcal{R}$ accounts for a non-negative acceleration factor a:

$$(\vec{x}, \vec{x}') \in G_r \text{ if and only if } \exists a \geq 0.\ \forall k : 0 \leq k < a.\ (\vec{x} + k \cdot r.\vec{u}, \vec{x}') \in \mathsf{Guard}_r,$$
$$\vec{x}'|_\Pi = \vec{x}|_\Pi, \vec{x}'|_\Gamma = \vec{x}|_\Gamma + a \cdot r.\vec{u}, \text{ and } \vec{x}'|_\mathcal{L} = \vec{x}|_\mathcal{L} + a \cdot (\vec{1}_{r.to} - \vec{1}_{r.from})$$

Discussions. If we ignore the location q_0, the counter automaton $\mathsf{CA}^1(A)$ has the same transition relation as the counter system of A; as defined in Section 2.2 or in [25], that is, with built-in single-rule acceleration. General acceleration procedures for reachability analysis were developed for counter automata [30, 4]. These techniques terminate on flat and flattable counter automata. A counter automaton is *flat*, if its control graph – built of locations and transitions – does not contain nested loops [10]. A counter automaton A is *flattable*, if there is a flat counter automaton F with the same reachability relation, that is, $\rightarrow_F^* = \rightarrow_A^*$. The counter automata $\mathsf{CA}^0(A)$ and $\mathsf{CA}^1(A)$ are obviously not flat, as can be seen from Figure 6, the question is whether they are flattable.

As can be seen from the definition of $\mathsf{CA}^1(A)$, single-rule acceleration has a special form: it merges successive occurrences of a rule of $\mathsf{CA}^0(A)$ into one transition, provided that the counter values are sufficiently large. The motivation behind this acceleration is to perform transitions of many processes in a distributed system in *parallel* [25], in contrast to compressing *sequential* steps.

The bounded diameter property for a threshold automaton A implies flattability of the counter automaton $\mathsf{CA}^0(A)$. It is sufficient to unroll $\mathsf{CA}^0(A)$ up to the diameter bound and add self-loops to model single-rule acceleration:

▶ **Proposition 21.** *For every unrestricted threshold automaton A, if the diameter of the counter system $CS(A)$ is bounded, then the counter automaton $\mathsf{CA}^0(A)$ is flattable.*

6 Flattability for non-canonical threshold automata

It is easy to see that the counter systems of non-canonical threshold automata do not have bounded diameter, when applying single-rule acceleration. Interestingly, we show that the respective counter automata for NCTA are flattable. Hence, they can be thus analyzed with general acceleration tools such as FAST [4].

Additional definitions. To prove flattability, we adapt a few definitions from [24]. Let $\mathcal{G} = \bigcup_{r\in\mathcal{R}} r.\Phi$. Then, $\Phi^{\mathrm{R}} = \{g \in \mathcal{G} \mid g \text{ is an upper guard}\}$ and $\Phi^{\mathrm{F}} = \{g \in \mathcal{G} \mid g \text{ is a lower guard}\}$. A *context* is a pair $(\Omega^{\mathrm{R}}, \Omega^{\mathrm{F}})$, where $\Omega^{\mathrm{R}} \subseteq \Phi^{\mathrm{R}}$ and $\Omega^{\mathrm{F}} \subseteq \Phi^{\mathrm{F}}$. The set Ω^{R} keeps track of unlocked guards from Φ^{R}, and the set Ω^{F} keeps track of locked guards from Φ^{F}. We usually denote a context with Ω, and refer to its first and second component by writing Ω^{R} and Ω^{F} respectively. For contexts Ω_1 and Ω_2, we say that $\Omega_1 \sqsubseteq \Omega_2$ if and only if $\Omega_1^{\mathrm{R}} \cup \Omega_1^{\mathrm{F}} \subseteq \Omega_2^{\mathrm{R}} \cup \Omega_2^{\mathrm{F}}$.

Finally, for a context Ω, we define a formula $form(\Omega)$ that summarizes the constraints of the guards that are locked/unlocked in the context: $\bigwedge_{\psi\in\Psi^+} \psi \wedge \bigwedge_{\psi\in\Psi^-} \neg\psi$ for $\Psi^+ = \Omega^{\mathrm{R}} \cup (\Phi^{\mathrm{F}} \setminus \Omega^{\mathrm{F}})$ and $\Psi^- = (\Phi^{\mathrm{R}} \setminus \Omega^{\mathrm{R}}) \cup \Omega^{\mathrm{F}}$. We write $[\![form(\Omega)]\!]$ to denote the set of vectors that satisfy $form(\Omega)$, that is, $\vec{x} \in [\![form(\Omega)]\!]$ if and only if $(\vec{x}|_\Gamma, \vec{x}|_\Pi) \models form(\Omega)$ holds true.

▶ **Definition 22.** For a NCTA $A = (\mathcal{L}, \mathcal{I}, \Gamma, \Pi, \mathcal{R})$ and a context Ω, we define the *slice* of A with context Ω as a threshold automaton $A|_\Omega = (\mathcal{L}, \mathcal{I}, \Gamma, \Pi, \mathcal{R}|_\Omega)$, where a rule $r \in \mathcal{R}$ belongs to $\mathcal{R}|_\Omega$ if and only if $form(\Omega) \rightarrow \bigwedge_{\varphi\in r.\Phi} \varphi$.

Overview of the proof. We start with an NCTA. The relation $\sqsubset$ is a partial order on the contexts. We construct a flat counter automaton as a composition of flat counter automata, one per context, that are then connected according to the partial order $\sqsubset$. Figure 7 sketches the construction. In more detail, for each context Ω of A we construct the slice. We show that when one removes the threshold guards from the slice, its counter automaton becomes structurally a BPP-net [16, 18], which are known to be flattable [30]. Thus, there is a

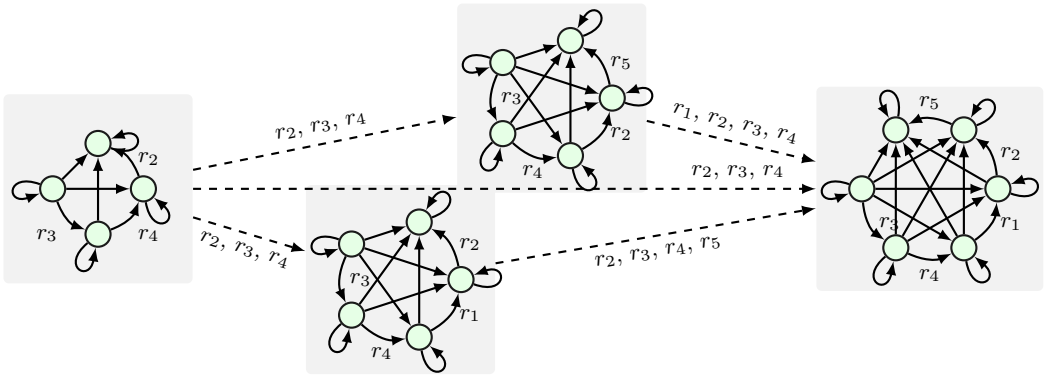

Figure 7 An example of the flattened threshold automaton from Figure 1. The edges connecting the gray blocks connect all the states inside the blocks.

flattened counter automaton $F(\Omega)$ for Ω. However, as $F(\Omega)$ does not have threshold guards, it allows transitions to leave the context Ω earlier than in the original counter system. Thus, we add additional constraints to $F(\Omega)$ to keep the transitions in the context, and form a "flat slice". Then, we combine flat slices for each context according to the partial order between the contexts, and obtain a flat counter automaton whose reachability relation is the same as of $\mathsf{CA}^0(A)$.

▶ **Proposition 23.** *For every non-canonical threshold automaton A and context Ω, there is a flat counter automaton $Flat(A|_\Omega)$ that has the same reachability relation when restricted to the* CA*-configurations that match the context, that is, $\rightarrow^*_{Flat(A|_\Omega)} \cap [\![form(\Omega)]\!]^2$ equals to $\rightarrow^*_{\mathsf{CA}^0(A)} \cap [\![form(\Omega)]\!]^2$.*

Assembling the flat counter automata for the slices. Fix a non-canonical threshold automaton $A = (\mathcal{L}, \mathcal{I}, \Gamma, \Pi, \mathcal{R})$. Proposition 23 allows us to flatten a single slice. To flatten $\mathsf{CA}^0(A)$, we flatten slices and connect them with context changing transitions.

As a first step, we enumerate all contexts $\Omega_1, \ldots, \Omega_K$, where $K = |\Phi^{\mathrm{R}} \times \Phi^{\mathrm{F}}|$. For each context $i \in \{1, \ldots, K\}$, we apply Proposition 23, to construct a flat counter automaton $Flat(i) = (Q_i, T_i, \mathsf{src}_i, \mathsf{tgt}_i, \{G^i_t\}_{t \in T_i})$. We assume that the sets $Q_1, \ldots, Q_K$ and $T_1, \ldots, T_K$ are all disjoint. We use $Flat(1), \ldots, Flat(K)$ to construct two sets of counter automata:

1. An automaton $FlatSlice(i)$ produces paths of $\mathsf{CA}^0(A)$ in the context Ω_i. Formally, $FlatSlice(i) = (Q_i, T_i, \mathsf{src}_i, \mathsf{tgt}_i, \{G^i_t \cap [\![form(\Omega)]\!]^2\}_{t \in T_i})$.
2. An automaton $Branch(i, j)$, for $1 \le i, j \le K$ such that $\Omega_i \sqsubseteq \Omega_j$ and $i \neq j$, produces the context-changing transitions from $FlatSlice(i)$ to $FlatSlice(j)$. Formally,

$$Branch(i,j) = (Q_i \cup Q_j, T_{i,j}, \mathsf{src}_{i,j}, \mathsf{tgt}_{i,j}, \{G^{i,j}_t\}_{t \in T_{i,j}}),$$

where the components of $Branch(i, j)$ are defined as follows for $t \in T_i$:

- There is a transition for each ith slice transition and jth slice state: $T_{i,j} = T_i \times Q_j$,
- The mappings are $\mathsf{src}_{i,j}((t, q)) = \mathsf{src}_i(t)$ and $\mathsf{tgt}_{i,j}((t, q)) = q$ for $q \in Q_j$, and
- We restrict the guards to the two contexts: $G^{i,j}_t = G^i_t \cap ([\![form(\Omega_i)]\!] \times [\![form(\Omega_j)]\!])$.

A flat version of $\mathsf{CA}^0(A)$ is the union of all flat slices and branches:

$$Flattened(A) = \bigcup_{1 \le i \le K} FlatSlice(i) \cup \bigcup_{(i,j) \in E} Branch(i,j) \text{ for } E = \{(i,j) \mid \Omega_i \sqsubseteq \Omega_j, i \neq j\} \quad (3)$$

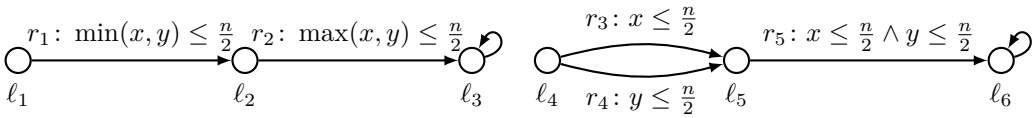

Figure 8 An unrestricted TA (left) and an equivalent threshold automaton (right).

We define the union $A \cup B$ as usual: The states, transitions, and flows of $A \cup B$ are the unions of the A's and B's states, transitions, and flows respectively. The source and target mappings are identical to the A's and B's mappings on their domains.

▶ **Theorem 24.** *For every non-canonical threshold automaton A, its flattened version has the same reachability relation:* $\rightarrow^*_{Flattened(A)} = \rightarrow^*_{CA^0(A)}$.

7 Conclusions

Verification of infinite-state systems and parameterized concurrent systems is a lively research area, e.g., see some recent results [19, 13, 1, 11, 17, 12, 31, 8, 2]. There are many different modeling frameworks, and it is not easy to understand relations between them. However, this understanding is of paramount importance for reusing existing tools. In this paper, on the one hand, we give reachability results for new classes of systems, and on the other hand, establish the relation of the model in [25, 21] to counter automata [10, 30]. We clarify the relation between the single rule acceleration introduced in [25] to acceleration in (flattable) counter automata [4, 30]. The single-rule acceleration in [25] is very simple compared to the general acceleration techniques [30, 4]. Still, it was demonstrated to be effective in parameterized verification of fault-tolerant distributed algorithms [22, 21].

The benefits of our extended framework are two-fold. On one hand, we can use our results to optimize threshold automata. Figure 8 shows an unrestricted threshold automaton that uses minimum and maximum. This UTA can be expressed as an equivalent threshold automaton by introducing more rules and guards (see Figure 8), which makes it harder to reason about. On the other hand, our framework permits some new guards, which have no corresponding encoding in threshold automata. For instance, a threshold $x < \sqrt{n/\log n}$ in [3] gives us such an example (though they are using the synchronous model of computation).

Some open questions still remain. Regarding application to distributed algorithms, we observe that in the pseudo code of several distributed consensus algorithms, processes pick the "*most often received value*" from a set of received values [5, 9]. A shared variable encoding – such as the one in [26] – maintains the number of messages with value 0 in a shared variable x_0, and the number of messages with value 1 in a shared variable x_1. The pseudo code statement about the "most often received value" needs a bounded difference guard "$x_1 - x_0 > 0$", which leads to undecidability as we show. This calls for further insights on modeling of such algorithms.

While we focused on reachability in this paper, as future work, we plan to lift the results of this paper to safety and liveness, following the ideas of [22].

References

1 Parosh Aziz Abdulla, Frédéric Haziza, and Lukás Holík. Parameterized verification through view abstraction. *STTT*, 18(5):495–516, 2016.

2 Benjamin Aminof, Sasha Rubin, Ilina Stoilkovska, Josef Widder, and Florian Zuleger. Parameterized model checking of synchronous distributed algorithms by abstraction. In *VMCAI*, volume 10747 of *LNCS*, pages 1–24, 2018.

3 Ziv Bar-Joseph and Michael Ben-Or. A tight lower bound for randomized synchronous consensus. In *Proceedings of the seventeenth annual ACM symposium on Principles of distributed computing*, pages 193–199. ACM, 1998.

4 Sébastien Bardin, Alain Finkel, Jérôme Leroux, and Laure Petrucci. Fast: acceleration from theory to practice. *STTT*, 10(5):401–424, 2008.

5 Martin Biely, Bernadette Charron-Bost, Antoine Gaillard, Martin Hutle, André Schiper, and Josef Widder. Tolerating corrupted communication. In *PODC*, pages 244–253, 2007.

6 Armin Biere, Alessandro Cimatti, Edmund M. Clarke, and Yunshan Zhu. Symbolic model checking without BDDs. In *TACAS*, volume 1579 of *LNCS*, pages 193–207, 1999.

7 Roderick Bloem, Swen Jacobs, Ayrat Khalimov, Igor Konnov, Sasha Rubin, Helmut Veith, and Josef Widder. *Decidability of Parameterized Verification*. Synthesis Lectures on Distributed Computing Theory. Morgan & Claypool Publishers, 2015.

8 Ahmed Bouajjani, Michael Emmi, Constantin Enea, and Jad Hamza. On reducing linearizability to state reachability. In *ICALP*, pages 95–107, 2015.

9 Bernadette Charron-Bost and André Schiper. The heard-of model: computing in distributed systems with benign faults. *Distributed Computing*, 22(1):49–71, 2009.

10 Hubert Comon and Yan Jurski. Multiple counters automata, safety analysis and Presburger arithmetic. In *CAV*, pages 268–279. Springer, 1998.

11 Giorgio Delzanno. A unified view of parameterized verification of abstract models of broadcast communication. *STTT*, 18(5):475–493, 2016.

12 Cezara Drăgoi, Thomas A. Henzinger, Helmut Veith, Josef Widder, and Damien Zufferey. A logic-based framework for verifying consensus algorithms. In *VMCAI*, volume 8318 of *LNCS*, pages 161–181, 2014.

13 Antoine Durand-Gasselin, Javier Esparza, Pierre Ganty, and Rupak Majumdar. Model checking parameterized asynchronous shared-memory systems. *Formal Methods in System Design*, 50(2-3):140–167, 2017.

14 Cynthia Dwork, Nancy Lynch, and Larry Stockmeyer. Consensus in the presence of partial synchrony. *J. ACM*, 35(2):288–323, 1988.

15 E.A. Emerson and K.S. Namjoshi. Reasoning about rings. In *POPL*, pages 85–94, 1995.

16 Javier Esparza. Petri nets, commutative context-free grammars, and basic parallel processes. *Fundam. Inform.*, 31(1):13–25, 1997.

17 Azadeh Farzan, Zachary Kincaid, and Andreas Podelski. Proving liveness of parameterized programs. In *LICS*, pages 185–196, 2016.

18 Laurent Fribourg and Hans Olsén. A decompositional approach for computing least fixed-points of datalog programs with z-counters. *Constraints*, 2(3/4):305–335, 1997.

19 Zeinab Ganjei, Ahmed Rezine, Petru Eles, and Zebo Peng. Counting dynamically synchronizing processes. *STTT*, 18(5):517–534, 2016.

20 Oscar H. Ibarra. Reversal-bounded multicounter machines and their decision problems. *J. ACM*, 25(1):116–133, 1978.

21 Igor Konnov, Marijana Lazić, Helmut Veith, and Josef Widder. Para^2: Parameterized path reduction, acceleration, and SMT for reachability in threshold-guarded distributed algorithms. *Formal Methods in System Design*, 2017.

22 Igor Konnov, Marijana Lazić, Helmut Veith, and Josef Widder. A short counterexample property for safety and liveness verification of fault-tolerant distributed algorithms. In *POPL*, pages 719–734, 2017.

23 Igor Konnov, Helmut Veith, and Josef Widder. On the completeness of bounded model checking for threshold-based distributed algorithms: Reachability. In *CONCUR*, volume 8704, pages 125–140. Elsevier, 2014.

24 Igor Konnov, Helmut Veith, and Josef Widder. SMT and POR beat counter abstraction: Parameterized model checking of threshold-based distributed algorithms. In *CAV (Part I)*, volume 9206 of *LNCS*, pages 85–102, 2015.

25 Igor Konnov, Helmut Veith, and Josef Widder. On the completeness of bounded model checking for threshold-based distributed algorithms: Reachability. *Information and Computation*, 252:95–109, 2017.

26 Igor Konnov, Josef Widder, Francesco Spegni, and Luca Spalazzi. Accuracy of message counting abstraction in fault-tolerant distributed algorithms. In *VMCAI*, pages 347–366, 2017.

27 Jure Kukovec. Generalizing threshold automata for reachability in parameterized systems. Master's thesis, University of Ljubljana, 2016. URL: `http://forsyte.at/wp-content/uploads/Kukovec-27142109-2016.pdf`.

28 Leslie Lamport. The part-time parliament. *ACM Trans. Comput. Syst.*, 16(2):133–169, 1998.

29 Marijana Lazić, Igor Konnov, Josef Widder, and Roderick Bloem. Synthesis of distributed algorithms with parameterized threshold guards. In *OPODIS*, volume 95 of *LIPIcs*, pages 32:1–32:20, 2017. `doi:10.4230/LIPIcs.OPODIS.2017.32`.

30 Jérôme Leroux and Grégoire Sutre. Flat counter automata almost everywhere! In *ATVA*, volume 5, pages 489–503. Springer, 2005.

31 Ognjen Maric, Christoph Sprenger, and David A. Basin. Cutoff bounds for consensus algorithms. In *CAV, Part II*, pages 217–237, 2017.

32 Marvin L Minsky. *Computation: finite and infinite machines*. Prentice-Hall, Inc., 1967.

33 Iulian Moraru, David G. Andersen, and Michael Kaminsky. There is more consensus in egalitarian parliaments. In *SOSP*, pages 358–372, 2013.

34 Brian M. Oki and Barbara Liskov. Viewstamped replication: A general primary copy. In *PODC*, pages 8–17, 1988.