Analyzing Guarded Protocols: Better Cutoffs, More Systems, More Expressivity

Swen Jacobs, Mouhammad Sakr

Reactive Systems Group, Saarland University, Germany
{jacobs,sakr}@react.uni-saarland.de

Abstract. We study cutoff results for parameterized verification and synthesis of guarded protocols, as introduced by Emerson and Kahlon (2000). Guarded protocols describe systems of processes whose transitions are enabled or disabled depending on the existence of other processes in certain local states. Cutoff results reduce reasoning about systems with an arbitrary number of processes to systems of a determined, fixed size. Our work is based on the observation that existing cutoff results for guarded protocols are often impractical, since they scale linearly in the number of local states of processes in the system. We provide new cutoffs that scale not with the number of local states, but with the number of guards in the system, which is in many cases much smaller. Furthermore, we consider generalizations of the type of guards and of the specifications under consideration, and present results for problems that have not been known to admit cutoffs before.

1 Introduction

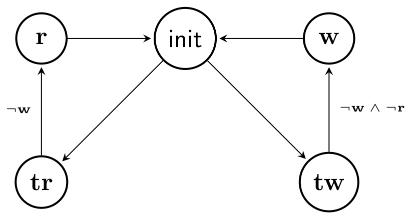
Concurrent systems are notoriously hard to get correct, and are therefore a promising application area for formal methods like model checking or synthesis. However, while such general-purpose formal methods can give strong correctness guarantees, they have two drawbacks: i) the state explosion problem prevents us from using them for systems with a large number of components, and ii) correctness properties are often expected to hold for an *arbitrary* number of components, which cannot be guaranteed without an additional argument that extends a proof of correctness to systems of arbitrary size. Both problems can be solved by approaches for *parameterized* model checking and synthesis, which give correctness guarantees for systems with any number of components without considering every possible system instance explicitly.

While the parameterized model checking problem (PMCP) is undecidable even if we restrict systems to uniform finite-state components [25], there exist several methods that decide the problem for specific classes of systems [2–4, 11, 15–17, 19, 21], many of which have been collected in surveys of the literature recently [9, 20]. Additionally, there are semi-decision procedures that are successful in a number of interesting cases [1, 10, 12, 23, 24]. In this paper, we consider the *cutoff* approach to the PMCP, that can guarantee properties of systems of arbitrary size by considering only systems of up to a certain fixed size. Thus, it provides a decision procedure for the PMCP if the model checking problem for a fixed number of components is decidable, e.g., if components are finite-state.

Guarded protocols, the systems under consideration, are composed of an arbitrary number of processes, each an instance of a finite-state process template. The process templates can be seen as synchronization skeletons [14], i.e., they only have to model the features of the system components that are important for their synchronization. Processes communicate by guarded updates, where guards are statements about other processes that are interpreted either conjunctively ("every other process satisfies the guard") or disjunctively ("there exists a process that satisfies the guard"). Conjunctive guards can be used to model synchronization by atomic sections or locks, while disjunctive guards can model pairwise rendezvous or token-passing.

Emerson and Namjoshi [18] have shown that the PMCP for systems that combine conjunctive and disjunctive guards is in general undecidable. Therefore, research in the literature has focused on systems that are restricted to one type of guard, called conjunctive or disjunctive systems, respectively. These classes of systems have been studied by Emerson and Kahlon [15,16], and cutoffs that depend on the size of process templates are known for specifications of the form $\forall \bar{p}.\ \Phi(\bar{p})$, where $\Phi(\bar{p})$ is an $\mathsf{LTL}\backslash\mathsf{X}$ property over the local states of one or more processes $\bar{p}$. Außerlechner et al. [6] have extended and improved these results, but a number of open issues remain. We will explain some of them in the following.

Motivating Example As an example, consider the reader-writer protocol on the right, modeling access to data shared between processes. A process that wants to read the data enters state tr ("try-read"). From tr, it can move to the reading state r. However, this transition is guarded by a statement $\neg w$. Formally, guards are sets of states, and $\neg w$ stands for the set of all states except w. Furthermore, this example is a conjunctive system, which means that a guard is interpreted as "all other processes have to be in the given set of states". Thus, to take the transition from tr to r, no other process should currently be in state w, i.e., writing the data. Similarly, a process that wants to enter w has to go through tw, but the transition into w is only enabled if no state is reading or writing.



For this example, consider parameterized safety conditions such as

$$\forall i \neq j.\, \mathsf{G}\left(\neg(w_i \wedge w_j) \wedge \neg(w_i \wedge r_j)\right),$$

where indices i and j refer to processes in the system. Emerson and Kahlon [15] show that properties from $\mathsf{LTL}\backslash\mathsf{X}$ of a single process have a cutoff of 2, which generalizes to a cutoff of $k+1$ for properties with k index variables. Moreover, they show that a cutoff linear in the size of the process template is sufficient to detect global deadlocks.

However, Außerlechner et al. [6] noted that for liveness properties such as

$$\forall i.\, \mathsf{G}\left((tr_i \rightarrow \mathsf{F}\, r_i) \wedge (tw_i \rightarrow \mathsf{F}\, w_i)\right),$$

an explicit treatment of fairness assumptions on the scheduling of processes is necessary. They show that the cutoff for $\mathsf{LTL}\backslash\mathsf{X}$ properties also holds under fairness assumptions, but a second aspect has to be considered to adequately treat liveness properties in guarded protocols: the detection of local deadlocks, i.e., whether some process stops after finite time. For this problem, they give a cutoff that is linear in the size of the process template, but a major restriction is that the cutoff only supports systems with 1-conjunctive guards, i.e., where each guard can only exclude a single state. Note that the example above is not supported by these results, since one of the guards excludes 2 states.

Another drawback of the existing results is that they use only minimal knowledge about the process templates: their size and the interpretation of guards. As a result, many cutoffs depend directly on the size of the process template. Intuitively, the *communication* between processes should be more important for the cutoff than their internal state space. This can also be seen in the example above: out of the 5 states, only 2 can be observed by the other processes, and can thus influence their behavior. In this paper, we will explore the idea of cutoffs that depend on the number and form of guards in the system.

Contributions We provide new cutoff results for guarded protocols:

1. For conjunctive systems, we extend the class of process templates that are supported by cutoff results, providing cutoff results for local deadlock detection in classes of templates that are not 1-conjunctive, and include examples such as the one above. However, we do not solve the general problem, and instead show that a cutoff for arbitrary conjunctive systems has to be at least quadratic in the size of the template.
2. For both conjunctive and disjunctive systems, we show that by closer analysis of process templates, in particular the number and the form of transition guards, we can obtain smaller cutoffs in many cases. This circumvents the tightness results of Außerlechner et al. [6], which state that no smaller cutoffs can exist for the class of all processes of a given size.
3. For disjunctive systems, we additionally extend both the class of process templates and the class of specifications that are supported by cutoff results. We show that systems with finite conjunctions of disjunctive guards are also supported by variations of the existing proof methods, and obtain cutoff results for these systems. Furthermore, we give cutoffs that support checking the simultaneous reachability (and repeated reachability) of a target set by *all* processes in a disjunctive system.

2 Preliminaries

2.1 System Model

In the following, let Q be a finite set of states.

Processes. A *process template* is a transition system $U = (Q_U, \mathsf{init}_U, \delta_U)$ with[1]

- $Q_U \subseteq Q$ is a finite set of states including the initial state init_U,
- $\delta_U : Q_U \times \mathcal{P}(Q) \times Q_U$ is a guarded transition relation.

Define the size of U as $|U| = |Q_U|$. An instance of template U will be called a *U-process.*

Guarded Protocols. Fix process templates A and B. A *guarded protocol* is a system $A \| B^n$, consisting of one A-process and n B-processes in an interleaving parallel composition.[2] We assume that $Q = Q_A \dot\cup Q_B$. Different B-processes are distinguished by subscript, i.e., for $i \in [1..n]$, B_i is the ith instance of B, and q_{B_i} is a state of B_i. A state of the A-process is denoted by q_A. We denote the set $\{A, B_1, \ldots, B_n\}$ as $\mathcal{P}$, and write p for a process in $\mathcal{P}$. For $U \in \{A, B\}$, we write G_U for the set of non-trivial guards that are used in δ_U, i.e., guards different from Q and $\emptyset$. Then, let $G = G_A \cup G_B$.

Disjunctive and Conjunctive Systems. In a guarded protocols, a local transition $(q_p, g, q'_p) \in \delta_U$ of p is *enabled in* s if its *guard* g is satisfied for p in s, written $(s, p) \models g$. We distinguish two types of guarded protocols, depending on their *interpretation of guards*:

$$\text{In disjunctive systems: } (s,p) \models g \quad \text{iff} \quad \exists p' \in \mathcal{P} \setminus \{p\} : \quad q_{p'} \in g.$$
$$\text{In conjunctive systems: } (s,p) \models g \quad \text{iff} \quad \forall p' \in \mathcal{P} \setminus \{p\} : \quad q_{p'} \in g.$$

Let $\mathsf{set}(s) = \{q_A, q_{B_1}, \ldots, q_{B_n}\}$, and for a set of processes $P = \{p_1, \ldots, p_k\}$, let $\mathsf{set}_P(s) = \{q_{p_1}, \ldots, q_{p_k}\}$. Then for disjunctive systems, we can more succinctly state that $(s, p) \models g$ iff $\mathsf{set}_{\mathcal{P} \setminus p}(s) \cap g \neq \emptyset$, and for conjunctive systems $(s, p) \models g$ iff $\mathsf{set}_{\mathcal{P} \setminus p}(s) \subseteq g$. A process is *enabled* in s if at least one of its transitions is enabled in s, otherwise it is *disabled.*

Like Emerson and Kahlon [15], we assume that in conjunctive systems init_A and init_B are contained in all guards, i.e., they act as neutral states. For conjunctive systems, we call a guard *k-conjunctive* if it is of the form $Q \setminus \{q_1, \ldots, q_k\}$ for some $q_1, \ldots, q_k \in Q$. A state q is *k-conjunctive* if all non-trivial guards of transitions from q are k'-conjunctive with $k' \leq k$. A conjunctive system is *k-conjunctive* if every state is k-conjunctive.

Then, $A \| B^n$ is defined as the transition system $(S, \mathsf{init}_S, \Delta)$ with

- set of global states $S = (Q_A) \times (Q_B)^n$,
- global initial state $\mathsf{init}_S = (\mathsf{init}_A, \mathsf{init}_B, \ldots, \mathsf{init}_B)$,
- and global transition relation $\Delta \subseteq S \times S$ with $(s, s') \in \Delta$ iff s' is obtained from $s = (q_A, q_{B_1}, \ldots, q_{B_n})$ by replacing one local state q_p with a new local state q'_p, where p is a U-process with local transition $(q_p, g, q'_p) \in \delta_U$ and $(s, p) \models g$.

[1] In contrast to Außerlechner et al. [6], for simplicity we only consider closed process templates. However, our results extend to open process templates in the same way as explained there.

[2] By similar arguments as in Emerson and Kahlon [15], our results can be extended to systems with an arbitrary number of process templates.

Runs. A *path* of a system is a sequence of states $x = s_1, s_2, \dots$ such that for all $m < |x|$ there is a transition $(s_m, s_{m+1}) \in \mathit{\Delta}$ based on a local transition of some process p_m. We say that process p_m *moves* at *moment* m. A path can be finite or infinite, and a *maximal path* is a path that cannot be extended, i.e., it is either infinite or ends in a state where no transition is enabled.

A system *run* is a maximal path starting in the initial state. We say that a run is *initializing* if every process that moves infinitely often also visits its initial state init infinitely often.

Given a system path $x = s_1, s_2, \dots$ and a process p, the *local path* of p in x is the projection $x(p) = s_1(p), s_2(p), \dots$ of x onto local states of p. A local path $x(p)$ is a *local run* if x is a run.

Deadlocks and Fairness. A run is *globally deadlocked* if it is finite. An infinite run is *locally deadlocked* for process p if there exists m such that p is disabled for all $s_{m'}$ with $m' \geq m$. A run is *deadlocked* if it is locally or globally deadlocked. A system *has a (local/global) deadlock* if it has a (locally/globally) deadlocked run. Note that absence of local deadlocks for all p implies absence of global deadlocks, but not the other way around.

A run $s_1, s_2, \dots$ is *unconditionally fair* if every process moves infinitely often. A run is *strong fair* if it is infinite and for every process p, if p is enabled infinitely often, then p moves infinitely often.

Remark. We consider these different notions of fairness for the following reason: we are interested in unconditionally fair runs of the system, which requires an assumption about scheduling. However, directly assuming unconditional fairness is too strong, since any run with a local deadlock will violate the assumption, and therefore satisfy the overall specification. Thus, we consider strong fairness as an assumption on the scheduler, and absence of local deadlocks as a property of the system that has to be proved. Together, they imply unconditional fairness.

2.2 Specifications

We consider formulas in $\mathsf{LTL}\backslash\mathsf{X}$, i.e., LTL without the next-time operator X. Let $h(A, B_{i_1}, \dots, B_{i_k})$ be an $\mathsf{LTL}\backslash\mathsf{X}$ formula over atomic propositions from Q_A and indexed propositions from $Q_B \times \{i_1, \dots, i_k\}$. For a system $A \| B^n$ with $n \geq k$ and $i_j \in [1..n]$, satisfaction of $\mathsf{A}\, h(A, B_{i_1}, \dots, B_{i_k})$ and $\mathsf{E}\, h(A, B_{i_1}, \dots, B_{i_k})$ is defined in the usual way (see e.g. Baier and Katoen [7]).

Parameterized Specifications. A *parameterized specification* is a temporal logic formula with indexed atomic propositions and quantification over indices. A *k-indexed formula* is of the form $\forall i_1, \dots, i_k.\, \mathsf{A}\, h(A, B_{i_1}, \dots, B_{i_k})$ or $\forall i_1, \dots, i_k.\, \mathsf{E}\, h(A, B_{i_1}, \dots, B_{i_k})$. For given $n \geq k$, by symmetry of guarded protocols (cp. Emerson and Kahlon [15]) we have

$$A \| B^n \models \forall i_1, \dots, i_k.\, \mathsf{A}\, h(A, B_{i_1}, \dots, B_{i_k}) \quad \text{iff} \quad A \| B^n \models \mathsf{A}\, h(A, B_1, \dots, B_k).$$

The latter formula is denoted by $\mathsf{A}\, h(A, B^{(k)})$, and we often use it instead of the original $\forall i_1, \dots, i_k.\, \mathsf{A}\, h(A, B_{i_1}, ..., B_{i_k})$. For formulas with path quantifier E, satisfaction is defined analogously, and equivalent to satisfaction of $\mathsf{E}\, h(A, B^{(k)})$.

2.3 Model Checking Problems and Cutoffs

For a given system $A\|B^n$ and specification $h(A, B^{(k)})$ with $n \geq k$,

- the *model checking problem* is to decide whether $A\|B^n \models \mathsf{A}\, h(A, B^{(k)})$,
- the (global/local) *deadlock detection problem* is to decide whether $A\|B^n$ has (global/local) deadlocks,
- the *parameterized model checking problem* (PMCP) is to decide whether $\forall m \geq n: \ A\|B^m \models \mathsf{A}\, h(A, B^{(k)})$, and
- the *parameterized (local/global) deadlock detection problem* is to decide whether for some $m \geq n$, $A\|B^m$ does have (global/local) deadlocks.

These definitions can be flavored with different notions of fairness, and with the E path quantifier instead of A. According to our remarks about fairness above, we are interested in proving the absence of local deadlocks under the assumption of strong fairness, which implies unconditional fairness and therefore allows us to separately prove the satisfaction of a temporal logic specification under the assumption of unconditional fairness.

Corresponding problems for the *synthesis* of process templates can be defined (compare Außerlechner et al. [6]). Parameterized synthesis based on cutoffs [22] is also supported by our cutoff results, but the details will not be necessary for understanding the results presented here.

Cutoffs. We define cutoffs with respect to a class of systems (either disjunctive or conjunctive), a class of process templates T, and a class of properties, which can be k-indexed formulas for some $k \in \mathbb{N}$ or the existence of (local/global) deadlocks. A *cutoff* for a given class of properties and a class of systems with processes from T is a number $c \in \mathbb{N}$ such that for all $A, B \in T$, all properties φ in the given class, and all $n \geq c$:

$$A\|B^n \models \varphi \ \Leftrightarrow \ A\|B^c \models \varphi.$$

Like the problem definitions above, cutoffs may additionally be flavoured with different notions of fairness.

Cutoffs and Decidability. Note that the existence of a cutoff implies that the parameterized model checking and parameterized deadlock detection problems are *decidable* iff their non-parameterized versions are decidable.

3 New Cutoff Results for Conjunctive Systems

In this section, we state our new results for conjunctive systems, and compare them to the previously known results in Table 1. We give improved cutoffs for global deadlock detection in general (Section 3.1), and for local deadlock detection for the restricted case of 1-conjunctive systems (Section 3.2). After that, we explain why local deadlock detection in general is hard, and identify a number of cases where we can solve the problem even for systems that are not 1-conjunctive (Sections 3.3 and 3.4). We do not improve on the cutoffs for $\mathsf{LTL}\backslash\mathsf{X}$ properties, since they are already very small and only depend on the number of index variables in the specification.

Additional Definitions. To analyze deadlocks in a given conjunctive system $A \| B^n$, we introduce additional definitions. A *deadset* is a minimal set D of local states of other processes that block all outgoing transitions of one process in its current state q. Formally, we say that $D \subseteq Q$ is a *deadset* of $q \in Q$ if:

i) $\forall (q, g, q') \in \delta : \exists q'' \in D : q'' \notin g$,
ii) D contains at most one state from Q_A, and
iii) there is no D' that satisfies i) and ii) with $D' \subset D$.

For a given local state q, $dead^{\wedge}_{q}$ is the set of all deadsets of q:

$$dead^{\wedge}_{q} = \{D \subseteq Q \mid D \text{ is a deadset of } q\}.$$

If $dead^{\wedge}_{q} = \emptyset$, then we say q is *free*. If a state q does not appear in $dead^{\wedge}_{q'}$ for any $q' \in Q$, then we say q is *non-blocking*. If a state q does not appear in $dead^{\wedge}_{q}$, then we say q is *not self-blocking*.

For example, in a system where B is the process template from Sect. 1, we have $dead^{\wedge}_{tw} = \{\{w\}, \{r\}\}$ and $dead^{\wedge}_{tr} = \{\{w\}\}$. For all other states $q \in \{\mathsf{init}, r, w\}$, we have $dead^{\wedge}_{q} = \emptyset$. Regardless of A, none of the deadsets contain a state from A, since the guards of B do not mention states of A.

In these terms, a globally deadlocked run is a run that ends in a global state s such that for every process p and its local state q, some $D \in dead^{\wedge}_{q}$ is contained in $\mathsf{set}(s)$. Similarly, a locally deadlocked run is a run such that one process p will eventually always remain in state q, and from some point on, we always have $D \subseteq \mathsf{set}(s)$ for some $D \in dead^{\wedge}_{q}$. Note that in this case, it can happen that there does not exist a single deadset D that is contained in $\mathsf{set}(s)$ all the time, but the run may *alternate* between different deadsets of q that are contained in $\mathsf{set}(s)$ at different times.

3.1 Global Deadlock Detection

For global deadlock detection, we show how to obtain improved cutoffs based on the number of free, non-blocking, and not self-blocking states in a given process template.

Theorem 1. *For conjunctive systems and process templates A, B, let*

- $k_1 = |D_1|$, *where $D_1 \subseteq Q_B$ is the set of free states in B,*
- $k_2 = |D_2 \setminus D_1|$, *where $D_2 \subseteq Q_B$ is the set of non-blocking states in B, and*
- $k_3 = |D_3 \setminus (D_1 \cup D_2)|$, *where $D_3 \subseteq Q_B$ is the set of not self-blocking states in B.*

Then $2|B| - 2k_1 - 2k_2 - k_3$ is a cutoff for global deadlock detection.

Proof Sketch. In order to simulate a globally deadlocked run $x = s_0, s_1, \ldots, s_m$ of a large system by a run y in the cutoff system, by Emerson and Kahlon [15] the following is sufficient. We analyze the set of local states $q \in Q$ that are present in the final state s_m of x, and distinguish whether any such q appears once in s_m, or multiple times. If q appears once, we identify one local run of x that ends in q, and replicate it in the cutoff system. If q appears multiple times, we do the same for two local runs of x that end in q. This construction ensures that in the resulting global run $x' = s'_0, \ldots, s'_m$ of the cutoff system, for any point in time t and any process p, we have $\mathsf{set}_p(s'_t) \subseteq \mathsf{set}_p(s_t)$. Therefore, all transitions in x' will be enabled, and x' is deadlocked in s'_m. If x' does not contain all local runs of x then there are stuttering steps in x', where no process moves. By removing these stuttering steps, we obtain the desired run y.

The construction of Emerson and Kahlon assumes that in the worst case all local states of B appear in the deadlocked state s_m. However, if $D_1 \subseteq Q_B$ are *free* local states, then we know that no state from D_1 can ever appear in s_m, and thus the cutoff is reduced by $2|D_1|$. Similarly, if $D_2 \subseteq Q_B$ are *non-blocking* states, then we know that no state from D_2 can be necessary for the deadlock in s_m, and therefore the construction will also work if we remove the local runs ending in D_2. This also reduces the cutoff by $2|D_2|$. Moreover, the original construction assumes that all local states q may be self-blocking, which requires the second local run that ends in q. If we know that $D_3 \subseteq Q_B$ are *not self-blocking*, then we only need one local run for each of these states, reducing the cutoff by $|D_3|$. If we combine all three cases, we get the statement of the theorem. □

Note that the sets of free, non-blocking, and not self-blocking states can be identified by a simple analysis of a single process template, and the cost of this analysis is negligible compared to the cost of a higher cutoff in verification of the system.

3.2 Local Deadlock Detection in 1-conjunctive Systems

For local deadlock detection, we first show that smaller cutoffs can be found by taking into account the transitions and guards of the process template. For a 1-conjunctive process template $U \in \{A, B\}$, let $G_{U,B}$ be the set of guards of U that exclude one of the states of B, i.e., that are of the form $g = Q \setminus \{q\}$ for some $q \in Q_B$. Furthermore, let $\mathsf{maxD}_U = max\{|D \cap Q_B| \mid D \in dead^{\wedge}_q \text{ for some } q \in Q_U\}$ be the maximal number of states from B that appear in any deadset of a state in U.

Theorem 2. *For conjunctive systems with process templates A, B, if process template $U \in \{A, B\}$ is 1-conjunctive, then the following are cutoffs for local deadlock detection in a U-process in non-fair runs:*

- $\mathsf{maxD}_U + 2$, *and*
- $|G_{U,B}| + 2$.

Proof Sketch. In order to simulate a locally deadlocked run $x = s_0, s_1, \ldots$ of a large system by a run y in the cutoff system, the following construction has been presented by Außerlechner et al. [5] for non-fair runs. Suppose process p is locally deadlocked in local state q after the system has entered state s_m. We first copy the local runs of A and p. Since the system is 1-conjunctive, every local state has a unique deadset. For each q' in the deadset D of q, we copy a local run from x that is in q' at time m, and modify it such that it stays in q' forever after this point in time. Thus, the process in q is locally deadlocked because all states in D will be present at any time after m. Finally, we copy one additional local run of a process that moves infinitely often in x. As in the proof of Theorem 1, all transitions of the resulting global run x' will be enabled, and we can obtain the desired run y by de-stuttering.

Note that the original proof uses one process for every state in the unique deadset D of the deadlocked local state q, and assumes that in the worst case we have $D \supseteq Q_B$, resulting in the cutoff of $|Q_B| + 2$ for all process templates with a given set of states Q_B. However, if we take into account the guards of transitions and the individual deadsets, we can obtain smaller cutoffs: in particular, instead of assuming that the size of some deadset is $|Q_B|$, we can compute the maximal size of actual deadsets maxD_U, and replace $|Q_B|$ by maxD_U to obtain a cutoff of $\mathsf{maxD}_U + 2$. Further, note that (since the system is 1-conjunctive) maxD_U is bounded by $|G_{U,B}|$, so $|G_{U,B}| + 2$ also is a cutoff. □

Theorem 3. *For conjunctive systems and process templates A, B, if process template $U \in \{A, B\}$ is 1-conjunctive, then $2|G_{U,B}|$ is a cutoff for local deadlock detection in a U-process in strong-fair runs.*

Proof Sketch. For fair runs, the construction by Außerlechner et al. [5] is similar as in the previous proof, but additionally we need to ensure that all processes either move infinitely often or are locally deadlocked. We explain the original construction in a new way that highlights our insight.

First, identify all states $q' \in Q_B$ in the deadset D of q such that there exists a locally deadlocked local run in x that eventually stays in q'. For each of these states, copy this local run. To ensure that these local runs are locally deadlocked also in the constructed run, add the states in their deadsets to D, and if q' is self-blocking then also copy another local run from x that eventually visits the state q' and stays there. Then repeat the procedure until no more states are added to D. Note that only states that are excluded in one of the (1-conjunctive) guards can be added to D, and for each state we have copied up to two local runs from x. Thus, the size of D is bounded by $|G_{U,B}|$, and in the worst case we have added $2|G_{U,B}|$ processes until now.

Then, let $D' \subseteq D$ be the set of states for which no process has been added thus far, and let m' be the time when all local runs that have been added until now are locally deadlocked. Copy for each of the states $q' \in D'$ one local run from x that is in q' at time m', and add a process that stays in init_B until time m'. Then after moment m' we can let all processes that are in D' move in the following way: (i) choose some $q' \in D'$ (ii) let the process that is in init_B move to q' (iii) let the process that was waiting in q' move to init_B (iv) repeat with fair choices of $q' \in D'$. Since each of these states must appear in x at any time after m' without a process being locally deadlocked in the state, there must be a local path from this state to itself in one of the local runs in x. Since for fair conjunctive systems we assume that they are initializing, this path must go through init_B, and the construction is guaranteed to work.

Note that overall, for each state in D we have copied either one or two local runs from x, so the bound for the number of these processes is still $2|G_{U,B}|$. Also note that the additional process that waits in init_U is only needed if at least one of the other processes is not locally deadlocked, thus it does not increase the needed number of processes. Finally, for the original locally deadlocked process we can distinguish two cases: i) if we have added $2|G_{U,B}|$ processes thus far, then the original process is deadlocked in a state that does not block any transition, and we can remove it since the run will exhibit a local deadlock regardless, or ii) if this is not the case, then even with the original process we need at most $2|G_{U,B}|$ processes overall. □

Note that in a 1-conjunctive process template U, we have $|G_{U,B}| \leq |Q_B| - 1$. Thus, our new cutoffs are always smaller or equal to the known cutoff from Außerlechner et al. [6].

Table 1: Cutoff Results for Conjunctive Systems

		EK [15]	AJK [6]	our work
k-indexed LTL\X	non-fair	$k+1$	$k+1$	unchanged
k-indexed LTL\X	fair	-	$k+1$	unchanged
Local Deadlock	non-fair	-	$\lvert B\rvert + 1^*$	$\mathsf{maxD}_U + 2$ and $\lvert G_{U,B}\rvert + 2^*$
Local Deadlock	fair	-	$2\lvert B\rvert - 2^{**}$	$2\lvert G_{U,B}\rvert^{**}$
Global Deadlock		$2\lvert B\rvert + 1$	$2\lvert B\rvert - 2$	$2\lvert B\rvert - 2k_1 - 2k_2 - k_3$

* : systems need to have alternation-bounded local deadlocks (see Sect. 3.4)
** : systems need to be initializing and have alternation-bounded local deadlocks
k_1: number of free states
k_2: number of non-blocking states that are not free
k_3: number of not self-blocking states that are not free or non-blocking

3.3 Local Deadlock Detection: Beyond 1-conjunctive Systems

While Theorems 2 and 3 improve on the local deadlock detection cutoff for conjunctive systems in some cases, the results are still restricted to 1-conjunctive process templates. The reason for this restriction is that when going beyond 1-conjunctive systems, the local deadlock detection cutoff (even without considering fairness) can be shown to grow at least quadratically in the number of states or guards, and it becomes very hard to determine a cutoff.

To analyze these cases, define the following: given a process template $U \in \{A, B\}$, a sequence of local states $q_1, \ldots, q_k$ is *connected* if $\forall q_i \in \{q_1, \ldots, q_k\} : \exists (q_i, g_i, q_{i+1}) \in \delta_U$. A *cycle* is a connected sequence of states $q, q_1, \ldots, q_k, q$ such that $\forall q_i, q_j \in \{q_1, \ldots, q_k\} : q_i \neq q_j$. We denote such a cycle by C_q. By abuse of notation, C_q is also used for the set of states on C_q. We denote the set of guards of the transitions on C_q as G_{C_q}.

Example 1. If we consider the process template in Figure 1 without the dashed parts, then it exhibits a local deadlock in state q_0 for 9 processes, but not for 8 processes: one process has to move to q_0, which has four deadsets: $\{a, c\}$, $\{a, d\}$, $\{b, c\}$, and $\{b, d\}$. To preserve a deadlock in q_0, the processes need to alternate between different deadsets while always at least covering one of them. To achieve this, for each cycle that starts and ends in states a, b, c, d, we need 2 processes that move along the cycle to keep all guards of q_0 covered at all times. Intuitively, one process per cycle has to be in the state of interest, or ready to enter it, and the other process is traveling on the cycle, waiting until the guards are satisfied.

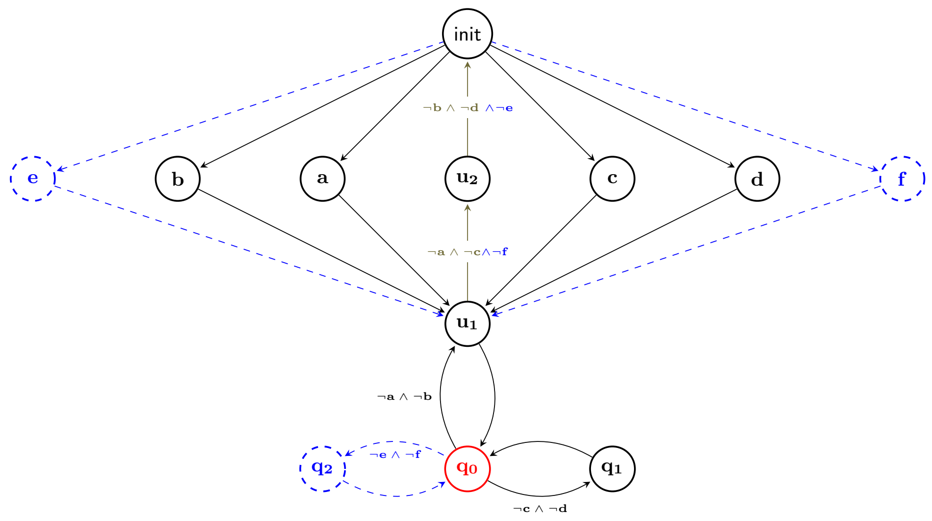


Fig. 1: Process Template with Quadratic Cutoff for Local Deadlocks

Now, consider the modified template (including the dashed parts) where we i) add two states e, f in a similar way as a, b, c, d, ii) add a new state connected to q_0 with guard $\neg e \wedge \neg f$, and iii) change the guards in the sequence from u_1 to init to $\neg a \wedge \neg c \wedge \neg e$ and $\neg b \wedge \neg d \wedge \neg f$, respectively. Then we have 6 cycles that need 2 processes each, and we need 13 processes to reach a local deadlock in q_0.

Moreover, consider the modified template where we increase the length of the path from u_1 to init by adding states u_3 and u_4, such that we obtain a sequence $(u_1, u_2, u_3, u_4, \mathsf{init})$, where transitions alternate between the two guards from the original sequence. Then, for every cycle we need 3 processes instead of 2, as otherwise they cannot traverse the cycle fast enough to ensure that the local deadlock is preserved infinitely long. That is, the template with both modifications now needs 19 processes to reach a local deadlock.

Observe that by increasing the height of the template, we increase the necessary number of processes without increasing the number of different guards. Moreover, when increasing both the width and height of the template, the number of processes that are necessary for a local deadlock increases quadratically with the size of the template.

This example leads us to the following result.

Theorem 4. *A cutoff for local deadlock detection for the class of all conjunctive systems must grow at least quadratically in the number of states. Furthermore, it cannot be bounded by the number of guards at all.*

Proof Sketch. For a system that does exhibit a local deadlock for some size n, but not for $n-1$, the cutoff cannot be smaller than n. Thus, the example shows that a cutoff for local deadlock detection in general is independent of the number of guards, and must grow at least quadratic in the size of the template. $\square$

Cutoffs that can in the best case be bounded by $|B|^2$ will not be very useful in practice. Therefore, instead of solving the general problem, we identify in the following a number of cases where the cutoff remains linear in the number of states or guards.

3.4 Systems with Alternation-bounded Local Deadlocks

When comparing the proof of Theorem 2 to Example 1, we note that the reason that the cutoff in Theorem 2 does not apply is the following: while in 1-conjunctive systems every state has a unique deadset, in the general case a state may have many deadsets, and the structure of the process template may require infinitely many alternations between different deadsets to preserve the local deadlock. Moreover, as shown in the example, the number of processes needed to alternate between deadsets may increase with the size of the template, even if the set of guards (and thus, the number of different deadsets) remains the same.

However, we can still obtain small cutoffs in some cases, based on the following observation: even if states have multiple deadsets, an infinite alternation between them may not be necessary to obtain a local deadlock. In the following, we will first show that for systems where infinite alternation between different deadsets is not necessary, the cutoff for 1-conjunctive systems applies, and then give a number of sufficient conditions to identify such systems.

Alternation-bounded Local Deadlocks. We say that a run $x = s_0, s_1, \ldots$ where process p is locally deadlocked in state q *is alternation-bounded* if there is a moment m and a single set $D \subseteq Q_B$ such that for all $m' > m$: $D \subseteq \mathsf{set}_{\mathcal{P}\setminus q}(s_{m'})$ and for some $q_A \in Q_A$, $D \cup q_A$ is a deadset of q. Intuitively, this means the B-states in the deadset that preserves the deadlock only change finitely often.

For $q \in Q$, we say that q *has alternation-bounded local deadlocks for* $c \in \mathbb{N}$ if the following holds for all $n \geq c$:

if $\quad A\|B^n$ has a local deadlock in q
then $A\|B^n$ has an alternation-bounded local deadlock in q.

Theorem 5. *For conjunctive systems and process templates A, B, the cutoffs of Theorem 2 apply for non-fair runs, and the cutoff of Theorem 3 applies for strong-fair runs if every $q \in Q$ has alternation-bounded local deadlocks for the cutoff value. In particular, this implies that the parameterized local deadlock detection problem is decidable.*

Proof Sketch. Suppose in run x of $A\|B^n$, with n greater than the cutoff value, process p is locally deadlocked in local state $q \in Q$, and q has alternation-bounded local deadlocks. Then there exists an alternation-bounded run x' of $A\|B^n$ in which p is locally deadlocked in q. That is, either the local deadlock in x' eventually is preserved by a sequence of deadsets with unique restriction to B-states, or a number of processes that is bounded by the size of the largest deadset is sufficient to preserve the local deadlock in q. In the latter case, we are done. In the former case, based on the set D, the run x' can be simulated with the same constructions as in the proofs of Theorems 2 and 3. $\square$

Sufficient Conditions for Alternation-bounded Local Deadlocks. In the following, we will identify four sufficient conditions that imply that a state q has alternation-bounded local deadlocks, and that can easily be checked directly on the process template.

Effectively 1-conjunctive states. We say that a state q is *effectively* 1*-conjunctive* if it is either 1-conjunctive or free.

Lemma 1. *If $q \in Q$ is effectively* 1*-conjunctive, then it has alternation-bounded local deadlocks for $c = 1$.*

Proof Sketch. If q is 1-conjunctive, then it has alternation-bounded local deadlocks since it has only a single deadset. If q is free, then a local deadlock in q is not possible, so the condition holds vacuously. $\square$

In the reader-writer example from Section 1, all states except tw are effectively 1-conjunctive.

Relaxing 1*-conjunctiveness.* For $q \in Q$, let G_q be the set of non-trivial guards in transitions from q. We say that state q is *relaxed* 1*-conjunctive* if G_q only contains guards of the form $Q \setminus \{q_1, \ldots, q_k\}$, where either

– at most one of the q_i is from Q_B, or
– whenever more than one q_i is from Q_B, then G_q must also contain a guard of the form $Q \setminus \{q'_1, \ldots, q'_k, q_i\}$ for one of these q_i and where all q'_j are from Q_A.

Lemma 2. *If $q \in Q$ is relaxed* 1*-conjunctive, then it has alternation-bounded local deadlocks for $c = 1$.*

Proof Sketch. Note that the guards we allow on transitions from a relaxed 1-conjunctive state each have at most one state from Q_B that can block the transition. Thus q does not necessarily have a unique deadset, but for each deadset D the restriction to states of B is unique. Thus, every run that is locally deadlocked in q will be alternation-bounded. □

Alternation-free. For a given state $q \in Q$, let D be the set of all local states that disable one of the k-conjunctive guards, with $k > 1$, on transitions from q. Then we say that q *is alternation-free* if the following condition holds for at most one $q' \in D$:

there exists a cycle $C_{q'} = q', \ldots, q' \in U$ with

- $q \notin C_{q'}$, and
- $\forall g \in G_{C_{q'}}\colon (q \in g \wedge \nexists g' \in G_q : g' \supseteq g)$.

Intuitively, this means that there is at most one state $q' \in D$ that is on a cycle that can be traversed while the local deadlock is preserved — and at least two such states would be needed to alternate between different deadsets.

In the reader-writer example from Section 1, state tw is alternation-free: i) $\{w, r\}$ is the set of states that disables the only guard that is not 1-conjunctive, and ii) all cycles that start and end in w contain also tw.

The following lemma directly follows from the explanation above.

Lemma 3. *If $q \in Q$ is alternation-free, then it has alternation-bounded local deadlocks for $c = 1$.*

Process templates with freely traversable lassos. While the three conditions above guarantee the existence of a fair alternation-bounded run if the original run was fair, the following condition in general returns a run that is not strong-fair. A *lasso* lo is a connected sequence of local states $q_0, \ldots, q_i, \ldots, q_k$ such that $q_0 = \mathsf{init}$ and $q_i, \ldots, q_k$ is a cycle. We denote by G_{lo} the set of guards of the transitions on lo. We say that a lasso lo is *freely traversable* with respect to a state $q \in Q$ if it does not contain q and for every deadset D of q, every $g \in G_{lo}$ contains $D \cup \{q\}$. Intuitively, these conditions ensure that lo can be executed after the system has reached any of the (minimal) deadlock configurations for q.

Lemma 4. *If there exists a freely traversable lasso in B with respect to $q \in Q$, then q has alternation-bounded local deadlocks in non-fair runs for $c = \mathsf{maxD}_U + 2$.*

Proof Sketch. Suppose there exists a freely traversable lasso with respect to q, and x is a run where process p is locally deadlocked in q, where p is not enabled anymore after time m. Then we obtain an alternation-bounded locally deadlocked run x' by picking a deadset D of q with $D \subseteq \mathsf{set}_{\mathcal{P}\setminus p}(x_m)$ and for every $q' \in D$ a local run from x that is in q' at time m. Since $n \geq c = \mathsf{maxD}_U + 2$, there is at least one other process in $A \| B^n$. We replace the local run of this process with a local run that stays in init_B until m, and after m is the only process that moves, along the freely traversable lasso we assumed to exist. Any further local runs stay in init_U forever. □

Theorem 5 and Lemmas 1 to 4 allow us to analyze the process templates, state by state, and to conclude the existence of a small cutoff for local deadlock detection in certain cases. The lemmas provide sufficient but not necessary conditions for the existence of alternation-bounded cutoffs. They provide a template for obtaining small cutoffs in certain cases, and for a given application they may be refined depending on domain-specific knowledge.

3.5 Local Deadlock Detection under Infinite Alternation

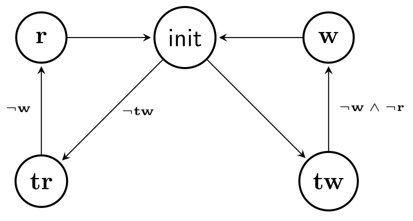
For systems that do not have alternation-bounded local deadlocks, it is very difficult to obtain cutoff results. For example, even in systems with a single 2-conjunctive and otherwise only 1-conjunctive guards, one can show that a cutoff based on the number of guards in general cannot exist. Moreover, the cutoff grows at least linearly in the number of states, or, more precisely, in the number of alternations between different deadsets that are necessary to traverse a cycle C_q for a state q from a deadset.

4 Verification of the Reader-Writer Example

We consider again the reader-writer example from Section 1, and show how our new results allow us to check correctness, find a bug, and check a fixed version.

With our results, we can for the first time check the given liveness property in a meaningful way, i.e., under the assumption of fair scheduling. Since all states in the process template have alternation-bounded local deadlocks for $c = 1$, by Theorems 3 and 5 the local deadlock detection cutoff for the system is $2|G_{B,B}| = 4$. No cutoff for this problem was known before. Moreover, compared to previous results we reduce the cutoff for global deadlock detection by recognizing that $k_1 = 3$ states can never be deadlocked, and $k_2 = 2$ additional states never appear in any guard. This reduces the cutoff to $2|B| - 2k_1 - 2k_2 = 10 - 6 - 4 = 0$, i.e., we detect that there are no global deadlocks without further analysis.

However, checking the system for local deadlocks shows that a local deadlock is possible: a process may forever be stuck in tw if the other processes move in a loop $(\mathsf{init}, tr, r)^\omega$ (and always at least one process is in r). To fix this, we can add an additional guard $\neg tw$ to the transition from init to tr, as shown in the process template to the right.



For the resulting system, our results give a local deadlock detection cutoff of $2|G_{B,B}| = 6$, and a global deadlock detection cutoff of $2|B| - 2k_1 - 2k_2 - k_3 = 10 - 6 - 2 - 1 = 1$ (where k_3 is the number of states that do appear in guards and could be deadlocked themselves, but do not have a transition that is blocked by another process in the same state).

5 New Cutoff Results for Disjunctive Systems

In this section, we state our new cutoff results for disjunctive systems, and compare them to the previously known results in Table 2. Moreover, we show two extensions of the class of problems for which cutoffs are available:

1. systems where transitions are guarded with a conjunction of disjunctive guards (Section 5.4), and
2. two important classes of specifications that cannot be expressed in prenex indexed temporal logic (Section 5.5).

To state our results, we need the following additional definitions. Fix process templates A, B with $G = G_A \cup G_B$. Let $|B|_G = |\{q \in Q_B \mid \exists g \in G : q \in g\}|$. For a state $q \in Q_B$ in a disjunctive system, define $\mathsf{Enable}_q = \{q' \in Q \mid \exists (q, g, q'') \in \delta_B : q' \in g\}$, i.e., the set of states of A and B that enable a transition from q.

5.1 Linear-Time Properties

Theorem 6. *For disjunctive systems, process templates A, B, and k-indexed properties Φ_k:*

- *$|G| + k + 1$ and $|B|_G + k + 1$ are cutoffs in non-fair runs,*
- *$|B|_G + |G| + k$ and $2|B|_G + k$ are cutoffs in unconditionally fair runs.*

Proof Sketch. Given a run x of $A \| B^n$ where $x(B_1), \dots, x(B_k)$ satisfy Φ_k, Emerson and Kahlon [15] showed how to construct a non-fair run y in the cutoff system that satisfies Φ_k. The run y includes the local runs $x(B_1), \dots, x(B_k)$, and additional runs that ensure that all the transitions are enabled: for every state $q \in Q_B$ that appears in x and the local run that first visits q, we add the prefix of that local run up to q, and then let it stay in q forever. One additional local run may have to be copied from x to ensure that the resulting run is infinite. Thus, the resulting cutoff is $|B| + k + 1$ in non-fair runs.

Based on an analysis of the process template B, we can find better cutoffs: as a first option, we can statically check which states do appear in a guard, and conclude that only those may need to be copied. This reduces the cutoff to $|B|_G + k + 1$. Furthermore, since our goal is to enable all transitions, it is also sufficient to only copy a local run for one *representative* state of each guard (the one that is visited first in x). In this way, we need at most one additional process per guard in B, i.e., $|G| + k + 1$ also is a cutoff for non-fair runs.

Außerlechner et al. [6] gave a construction that builds on the steps explained above, and additionally preserves unconditional fairness in a given run. To this end, distinguish local states that appear finitely often or infinitely often in x. If state q appears infinitely often, there must be a cycle $C_q = q, \dots, q$ in one of the local runs. To ensure fairness while always covering q, we add two copies of the shortest local path to q, and let the two processes take turns in moving through C_q (i.e., while one of them moves through C_q, the other one stays in q). If state q appears finitely often in x, we add a copy of the shortest path to

q, and identify the moment m_q when q appears for the last time in x. Instead of staying in q forever, we let the copied local run stay in q until m_q, and then move along the local path that leaves q at that time in x, until it reaches a state q' that appears infinitely often. From that point on, we let the process move in a fair way based on a cycle $C_{q'}$ taken from x. This original construction gives a cutoff of $2|B|+k-1$, since in the worst case all states appear infinitely often and we need two copies for each, but at least one of them must also appear infinitely often in the k processes that have to satisfy the specification.

Like in the non-fair case, an analysis of the template gives us better cutoffs. As a first approximation, we can again limit the construction to states in $|B|_G$, and obtain the cutoff $2|B|_G + k$ (now we can not assume that one of the states also appears in the k processes). Moreover, from the states in $|B|_G$ that appear infinitely often we can again chose one representative for each guard, and only add two local runs for each representative. This does not work for the processes that are visited finitely often, since we need to move them into an infinitely visited state to ensure fairness, and then need a different representative. To compute the cutoff, suppose f states from $|B|_G$ are visited finitely often, and i states infinitely often. From the latter, there are r states for which we added two local runs, with $r \leq |G|$ and $r \leq i$. Then we need at most $f + 2r + k$ local runs (including the k processes that satisfy the specification). However, we have $f \leq |B|_G - i$, and therefore $f + 2r + k \leq |B|_G - i + 2r + k \leq |B|_G + r + k \leq |B|_G + |G| + k$. □

5.2 Global Deadlock Detection

Let $\mathcal{N} = \{q \in Q_B \mid q \in \mathsf{Enable}_q\}$, and let $\mathcal{N}^*$ be the maximal subset (wrt. number of elements) of $\mathcal{N}$ such that $\forall q_i, q_j \in \mathcal{N}^* : q_i \notin \mathsf{Enable}_{q_j} \land q_j \notin \mathsf{Enable}_{q_i}$.

Theorem 7. *For disjunctive systems and process templates A, B, $|B|_G + |\mathcal{N}^*|$ is a cutoff for global deadlock detection.*

Proof Sketch. To construct a globally deadlocked run in the cutoff system, for each state from $\mathcal{N}$ that appears in the deadlock, we copy the according local run. To simulate the remaining part of x, we use the same construction as for fair runs in the proof of Thm. 6, except that local states that appear in the deadlock are considered to be visited infinitely often (and we don't need the fair extension of runs after reaching the state). Thus, the resulting run will be globally deadlocked, and all transitions up to the deadlock will be enabled. The number of local runs is bounded by $|\mathcal{N}| + f + i$, where i is the number of states from $|B|_G$ that appear in the deadlock and are not in $\mathcal{N}$, and f is the states from $|B|_G$ that appear in the run, but not in the deadlock. Since $f + i \leq |B|_G$ and $\mathcal{N}^*$ is the maximal subset of $\mathcal{N}$ that can appear together in a global deadlock, the number of needed local runs is bounded by $|\mathcal{N}^*| + |B|_G$. □

Remark. To compute $\mathcal{N}^*$ exactly, we need to find the smallest set of states in $\mathcal{N}$ that do not satisfy the additional condition. This amounts to finding the minimum vertex cover (MVC) for the graph with vertices from $\mathcal{N}$ and edges from q_i to q_j if $q_i \in \mathsf{Enable}_{q_j}$.

5.3 Local Deadlock Detection

Theorem 8. *For disjunctive systems and process templates A, B:*

- $m + |G| + 1$ *is a cutoff for local deadlock detection in non-fair runs, where* $m = \max_{q \in Q_B^*}\{|\mathsf{Enable}_q|\}$ *for* $Q_B^* = \{q \in Q_B \mid |\mathsf{Enable}_q| < |B|\}$,
- $|B|_G + |G| + 1$ *and* $2|B|_G + 1$ *are cutoffs for local deadlock detection in unconditionally fair runs.*

Proof Sketch. Based on what we have already shown, the fair case is simpler: we copy the local runs of A and the deadlocked process, and for the other processes use the same construction as in the fair case of Thm. 6. The local deadlock is preserved since states that appear finitely often in the original run also appear finitely often in the constructed run, and the cutoffs are $2|B|_G + 1$ and $|B|_G + |G| + 1$.

For the non-fair case, we use a combination of the constructions for the fair and non-fair case from Thm. 6: if in run x a process is locally deadlocked in local state q, then for states in Enable_q that appear in x we use the construction for finitely appearing states in fair runs. For the remaining states, we use the non-fair construction, i.e., we find one representative per guard and stay there forever, except that representatives now can never be from Enable_q. The construction ensures that all transitions that are taken are enabled, and eventually all transitions from q are disabled. Since m gives a bound on the number of states that can be in Enable_q, the cutoff we get is $m + |G| + 1$. □

Table 2: Cutoff Results for Disjunctive Systems

		EK [15]	AJK [6]	our work
k-indexed LTL\X	non-fair	$\lvert B\rvert + k + 1$	$\lvert B\rvert + k + 1$	$\lvert G\rvert + k + 1$ and $\lvert B\rvert_G + k + 1$
k-indexed LTL\X	fair	-	$2\lvert B\rvert + k - 1$	$\lvert B\rvert_G + \lvert G\rvert + k$ and $2\lvert B\rvert_G + k$
Local Deadlock	non-fair	-	$\lvert B\rvert + 2$	$m + \lvert G\rvert + 1$, with $m < \lvert B\rvert$
Local Deadlock	fair	-	$2\lvert B\rvert - 1$	$\lvert B\rvert_G + \lvert G\rvert + 1$ and $2\lvert B\rvert_G + 1$
Global Deadlock		-	$2\lvert B\rvert - 1$	$\lvert B\rvert + \lvert\mathcal{N}^*\rvert$ with $\lvert\mathcal{N}^*\rvert < \lvert B\rvert$

5.4 Systems with Conjunctions of Disjunctive Guards

We consider systems where a transition can be guarded by a set of sets of states, interpreted as a conjunction of disjunctive guards. I.e., a guard $\{D_1, \ldots, D_n\}$ is satisfied in a given global state if for all $i = 1, \dots, n$, there exists another process in a state from D_i.

We observe that for this class of systems, most of the original proof ideas still work. For results that depend on the number of guards, we have to count the number of different conjuncts in guards.

Theorem 9. *For systems with conjunctions of disjunctive guards, cutoff results for disjunctive systems that do not depend on the number of guards still hold (first and second column of results in Table 2, and cutoffs in the third column that only refer to $|B|_G$ and constants).*

Cutoff results that depend on the number of guards (last column of Table 2) hold if we consider the number of conjuncts in guards instead. For results that additionally refer to some measure of the sets of enabling states (m and $|\mathcal{N}^*|$, respectively), we obtain a valid cutoff for systems with conjunctions of disjunctive guards if we replace this measure by $|B| - 1$.

In particular, the existence of a cutoff implies that the respective PMCP and parameterized deadlock detection problems are decidable.

Proof Ideas. The cutoff results that are independent of the number of guards still hold since all of the original proof constructions still work. To simulate a run x of a large system in a run y the cutoff system, one task is to make sure that all necessary transitions are enabled in the cutoff system. The construction that is used to do this works for conjunctions of disjunctive guards just as well. By a similar argument, deadlocks are preserved in the same way as for disjunctive systems.

For cutoffs that depend on the number of guards, transitions with conjunctions of disjunctive guards require us to use one representative for each conjunct in a guard, in the construction explained in the proof of Theorem 6.

Finally, the reductions of the cutoff based on the analysis of states that can or cannot appear together in a deadlock do not work in these extended systems, and we have to replace m and $|\mathcal{N}^*|$ by $|B| - 1$ in the cutoffs. The reason is that Enable_q is now not a set of states anymore, but a set of sets of states. A more detailed analysis based on this observation may be possible, but is still open. $\square$

5.5 Simultaneous Reachability of Target States

An important class of properties for parameterized systems asks for the reachability of a global state where all processes of type B are in a given local state q (compare Delzanno et al. [13]). This can be written in indexed $\mathsf{LTL}\backslash\mathsf{X}$ as $\mathsf{F}\,\forall i.q_i$, but is not expressible in the fragment where index quantifiers have to be in prenex form. We denote this class of specifications as TARGET. Similarly, repeated reachability of q by all states simultaneously can be written $\mathsf{GF}\,\forall i.q_i$, and is also not expressible in prenex form. We denote this class of specifications as REPEAT-TARGET.

Theorem 10 (Disjunctive Target and Repeat-Target). *For disjunctive systems: $|B|$ is a cutoff for checking* TARGET *and* REPEAT-TARGET.

In particular, the PMCP with respect to Target *and* Repeat-Target *in disjunctive systems is decidable.*

Proof Ideas. We can simulate a run x in a large system where all processes are in q at time m in the cutoff system by first moving one process into each state that appears in x before m, in the same order as in x. To make all processes reach q, we move them out of their respective states in the same order as they have moved out of them in x. For this construction, we need at most $|B|$ processes.

If in x the processes are repeatedly in q at the same time, then we can simulate this also in the cutoff system: if $m' > m$ is a point in time where this happens again, then we use the same construction as above, except that we consider all states that are visited between m and m', and we move to these states from q instead from init. The correctness argument is the same, however.

Finally, if the run with Repeat-Target should be fair, then we do not select *any* m' with the property above, but we choose it such that all processes move between m and m'. If the original run x is fair, then such an m' must exist. $\square$

6 Conclusion

We have shown that better cutoffs for guarded protocols can be obtained by analyzing properties of the process templates, in particular the number and form of transition guards. We have further shown that cutoff results for disjunctive systems can be extended to a new class of systems with conjunctions of disjunctive guards, and to specifications Target and Repeat-Target, that have not been considered for guarded protocols before.

For conjunctive systems, previous works have treated local deadlock detection only for the restricted case of systems with 1-conjunctive guards. We have considered the general case, and have shown that it is very difficult — the cutoffs grow independently of the number of guards, and at least quadratically in the size of the process template. To circumvent this worst-case behavior, we have identified a number of conditions under which a small cutoff can be obtained even for systems that are not 1-conjunctive.

By providing cutoffs for several problems that were previously not known to be decidable, we have in particular proved their decidability.

Our work is inspired by applications in parameterized synthesis [8,22], where the goal is to automatically *construct* process templates such that a given specification is satisfied in systems with an arbitrary number of components. In this setting, deadlock detection and expressive specifications are particularly important, since *all* relevant properties of the system have to be specified.

Acknowledgements. We thank Ayrat Khalimov for fruitful discussions on guarded protocols, Martin Zimmermann for suggestions regarding a draft of this work, and the reviewers for insightful comments and suggestions. This work was supported by the German Research Foundation (DFG) through project Automatic Synthesis of Distributed and Parameterized Systems (JA 2357/2-1).

References

1. P. A. Abdulla, F. Haziza, and L. Holík. All for the price of few. In *VMCAI*, volume 7737 of *LNCS*, pages 476–495. Springer, 2013. doi:10.1007/978-3-642-35873-9_28.
2. B. Aminof, S. Jacobs, A. Khalimov, and S. Rubin. Parameterized model checking of token-passing systems. In *VMCAI*, volume 8318 of *LNCS*, pages 262–281. Springer, 2014. doi:10.1007/978-3-642-54013-4_15.
3. B. Aminof, T. Kotek, S. Rubin, F. Spegni, and H. Veith. Parameterized model checking of rendezvous systems. In *CONCUR*, volume 8704 of *LNCS*, pages 109–124. Springer, 2014. doi:10.1007/978-3-662-44584-6_9.
4. Benjamin Aminof and Sasha Rubin. Model checking parameterised multi-token systems via the composition method. In *IJCAR*, volume 9706 of *LNCS*, pages 499–515. Springer, 2016. doi:10.1007/978-3-319-40229-1_34.
5. Simon Außerlechner, Swen Jacobs, and Ayrat Khalimov. Tight cutoffs for guarded protocols with fairness. *CoRR*, abs/1505.03273, 2015. Extended version with full proofs. URL: http://arxiv.org/abs/1505.03273.
6. Simon Außerlechner, Swen Jacobs, and Ayrat Khalimov. Tight cutoffs for guarded protocols with fairness. In *VMCAI*, volume 9583 of *LNCS*, pages 476–494. Springer, 2016. doi:10.1007/978-3-662-49122-5_23.
7. Christel Baier and Joost-Pieter Katoen. *Principles of model checking*, volume 26202649. MIT press Cambridge, 2008.
8. Roderick Bloem, Swen Jacobs, and Ayrat Khalimov. Parameterized synthesis case study: AMBA AHB. In *SYNT*, volume 157 of *EPTCS*, pages 68–83, 2014. doi:10.4204/EPTCS.157.9.
9. 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. doi:10.2200/S00658ED1V01Y201508DCT013.
10. A. Bouajjani, B. Jonsson, M. Nilsson, and T. Touili. Regular model checking. In *CAV*, volume 1855 of *LNCS*, pages 403–418. Springer, 2000. doi:10.1007/10722167_31.
11. E. M. Clarke, M. Talupur, T. Touili, and H. Veith. Verification by network decomposition. In *CONCUR*, volume 3170 of *LNCS*, pages 276–291. Springer, 2004. doi:10.1007/978-3-540-28644-8_18.
12. E. M. Clarke, M. Talupur, and H. Veith. Proving ptolemy right: The environment abstraction framework for model checking concurrent systems. In *TACAS*, volume 4963 of *LNCS*, pages 33–47. Springer, 2008. doi:10.1007/978-3-540-78800-3_4.
13. Giorgio Delzanno, Arnaud Sangnier, and Gianluigi Zavattaro. Parameterized verification of ad hoc networks. In *CONCUR*, volume 6269 of *LNCS*, pages 313–327. Springer, 2010. doi:10.1007/978-3-642-15375-4_22.
14. E. A. Emerson and E. M. Clarke. Using branching time temporal logic to synthesize synchronization skeletons. *Sci. Comput. Program.*, 2(3):241–266, 1982. doi:10.1016/0167-6423(83)90017-5.
15. E. A. Emerson and V. Kahlon. Reducing model checking of the many to the few. In *CADE*, volume 1831 of *LNCS*, pages 236–254. Springer, 2000. doi:10.1007/10721959_19.
16. E. A. Emerson and V. Kahlon. Model checking guarded protocols. In *LICS*, pages 361–370. IEEE Computer Society, 2003. doi:10.1109/LICS.2003.1210076.

17. E. A. Emerson and K. S. Namjoshi. On reasoning about rings. *Foundations of Computer Science*, 14(4):527–549, 2003. doi:10.1142/S0129054103001881.
18. E. Allen Emerson and Kedar S. Namjoshi. Automatic verification of parameterized synchronous systems (extended abstract). In *CAV*, volume 1102 of *LNCS*, pages 87–98. Springer, 1996. doi:10.1007/3-540-61474-5_60.
19. J. Esparza, A. Finkel, and R. Mayr. On the verification of broadcast protocols. In *LICS*, pages 352–359. IEEE Computer Society, 1999. doi:10.1109/LICS.1999.782630.
20. Javier Esparza. Keeping a crowd safe: On the complexity of parameterized verification (invited talk). In *STACS*, volume 25 of *LIPIcs*, pages 1–10. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2014. doi:10.4230/LIPIcs.STACS.2014.1.
21. S. M. German and A. P. Sistla. Reasoning about systems with many processes. *J. ACM*, 39(3):675–735, 1992. doi:10.1145/146637.146681.
22. S. Jacobs and R. Bloem. Parameterized synthesis. *Logical Methods in Computer Science*, 10:1–29, 2014. doi:10.2168/LMCS-10(1:12)2014.
23. A. Kaiser, D. Kroening, and T. Wahl. Dynamic cutoff detection in parameterized concurrent programs. In *CAV*, volume 6174 of *LNCS*, pages 645–659. Springer, 2010. doi:10.1007/978-3-642-14295-6_55.
24. R. P. Kurshan and K. L. McMillan. A structural induction theorem for processes. *Inf. and Comp.*, 117(1):1–11, 1995. doi:10.1006/inco.1995.1024.
25. I. Suzuki. Proving properties of a ring of finite state machines. *Inf. Process. Lett.*, 28(4):213–214, 1988. doi:10.1016/0020-0190(88)90211-6.