The complexity of bounded context switching with dynamic thread creation

Pascal Baumann
Max Planck Institute for Software Systems (MPI-SWS), Germany
pbaumann@mpi-sws.org

Based on joint work with Rupak Majumdar, Ramanathan S. Thinniyam, and Georg Zetzsche accepted for ICALP 2020.

1 Introduction

There is a complexity gap between EXPSPACE and 2EXPSPACE that shows up in several problems in the safety verification of multithreaded programs.

Atig, Bouajjani, and Qadeer [1] study safety verification for *dynamic networks of concurrent pushdown systems* (DCPS), a theoretical model for multithreaded recursive programs with a finite shared global state, where threads can be recursive and can dynamically spawn additional threads. Unrestricted reachability is undecidable in this model. To ensure decidability, like many other works [13, 10, 12, 9], they assume a bound K that restricts each thread to have at most K context switches. For safety verification in this model, formulated as global state reachability, they show a lower bound of EXPSPACE and an upper bound of 2EXPSPACE, "closing the gap" is left open.

Kaiser, Kroening, and Wahl [7] study safety verification of multithreaded *non-recursive* programs with local and global Boolean variables. In this model, an arbitrary number of non-recursive threads execute over shared global state, but each thread can maintain local state in Boolean variables. Although their paper does not provide an explicit complexity bound, a lower bound of EXPSPACE and an upper bound of 2EXPSPACE can be derived from a reduction from Petri net coverability and their algorithm respectively.

Interestingly, when we restrict the models to disallow either context switches (i.e., each thread runs atomically to completion) or local state in the form of the pushdown stack or local variables (but allow arbitrary context switches), safety verification is in EXPSPACE [1, 6].

Thus, the complexity gap begs the question whether or not the combination of *local state* (maintained in local variables or in the stack) and bounded *context switching* provides additional power to computation. In this paper, we show that indeed it does. In fact, the combination of local state and just *one* context switch is sufficient to achieve 2EXPSPACE lower bounds for these problems. This closes the complexity gap.

We believe the constructions and models that we use along the way are of independent interest. We introduce *transducer-defined* Petri nets (TDPNs), a succinct representation for Petri nets. The places in a TDPN are encoded using words over a fixed alphabet, and the transitions are described by length-preserving transducers. We show that coverability for TDPNs is 2EXPSPACE-complete[1] and give a polynomial-time reduction from coverability for TDPNs to safety verification for DCPS with one context switch.

The idea of the latter reduction is to map a (compressed) place to the stack of a thread and a marking to the set of currently spawned threads. A key obstacle in the simulation is to "transfer" potentially exponential amount of information from before a transition to after

[1] After submitting this work to ICALP 2020, we were made aware of "(level 1) counter systems with chained counters" from [3], for which 2EXPSPACE-hardness of state reachability is shown in [3, Theorem 14]. The 2EXPSPACE-hardness of coverability in TDPN could also be deduced from that result.

it through a polynomial-sized global store. We present a "guess and verify" procedure, using non-determinism and the use of additional threads to verify a stack content letter-by-letter.

In order to show 2EXPSPACE-hardness for TDPNs, we introduce the model of *recursive net programs* (RNPs), which add the power of making possibly recursive procedure calls to the model of net programs (i.e., programs with access to Petri net counters). The addition of recursion enables us to replace the "copy and paste code" idea in Lipton's construction to show EXPSPACE-hardness of Petri net coverability [11] with a more succinct and cleaner program description where the copies are instead represented by different values of the local variables of the procedures. The net effect is to push the requirement for copies into the call stack of the RNP while maintaining a syntax which gives us a RNP which is polynomial in the size of a given counter program. When the stack size is bounded by an exponential function of the size of the program, we get a 2EXPSPACE-lower bound. We show that recursive net programs with exponentially large stacks can be simulated by TDPNs.

Finally, we note that the 2EXPSPACE lower bound holds for DCPS where each stack is bounded by a linear function of the size. Such stacks can be encoded by polynomially many local Boolean variables, giving us a 2EXPSPACE lower bound for the model of Kaiser et al.

In summary, we introduce a number of natural 2EXPSPACE-complete problems and, through a series of reductions, close an exponential gap in the complexity of safety verification for multithreaded recursive programs.

2 Preliminaries

A *multiset* $\mathbf{m}\colon S \to \mathbb{N}$ over a set S maps each element of S to a natural number. Let $\mathbb{M}[S]$ be the set of all multisets over S. We treat sets as a special case of multisets where each element is mapped onto 0 or 1. We sometimes write $\mathbf{m} = [\![a_1, a_1, a_3]\!]$ for the multiset $\mathbf{m} \in \mathbb{M}[S]$ such that $\mathbf{m}(a_1) = 2$, $\mathbf{m}(a_3) = 1$, and $\mathbf{m}(a) = 0$ for each $a \in S \backslash \{a_1, a_3\}$. The empty multiset is denoted $\emptyset$. The *size* of a multiset $\mathbf{m}$, denoted $|\mathbf{m}|$, is given by $\sum_{a \in S} \mathbf{m}(a)$. Note that this definition applies to sets as well.

Given two multisets $\mathbf{m}, \mathbf{m}' \in \mathbb{M}[S]$ we define $\mathbf{m} \oplus \mathbf{m}' \in \mathbb{M}[S]$ to be a multiset such that for all $a \in S$, we have $(\mathbf{m} \oplus \mathbf{m}')(a) = \mathbf{m}(a) + \mathbf{m}'(a)$. We also define the natural order $\preceq$ on $\mathbb{M}[S]$ as follows: $\mathbf{m} \preceq \mathbf{m}'$ iff there exists $\mathbf{m}^\Delta \in \mathbb{M}[S]$ such that $\mathbf{m} \oplus \mathbf{m}^\Delta = \mathbf{m}'$. We also define $\mathbf{m} \ominus \mathbf{m}'$ for $\mathbf{m}' \preceq \mathbf{m}$ analogously: for all $a \in S$, we have $(\mathbf{m} \ominus \mathbf{m}')(a) = \mathbf{m}(a) - \mathbf{m}'(a)$.

A *Dynamic Network of Concurrent Pushdown Systems (*DCPS*)* $\mathcal{A} = (G, \Gamma, \Delta, g_0, \gamma_0)$ consists of a finite set of *(global) states* G, a finite alphabet of *stack symbols* Γ, an *initial state* $g_0 \in G$, an *initial stack symbol* $\gamma_0 \in \Gamma$, and a finite set of *transition rules* Δ. Elements of Δ have one of the two forms (1) $g|\gamma \hookrightarrow g'|w'$, or (2) $g|\gamma \hookrightarrow g'|w' \triangleright \gamma'$, where $g, g' \in G$, $\gamma, \gamma' \in \Gamma$, $w' \in \Gamma^*$, and $|w'| \leq 2$. Rules of the first kind allow the DCPS to take a single step in one of the pushdown threads while the second additionally spawn a new thread with top of stack symbol γ'. The *size* of $\mathcal{A}$ is defined as $|\mathcal{A}| = |G| + |\Gamma| + |\Delta|$.

The set of *configurations* of $\mathcal{A}$ is $G \times (\Gamma^* \times \mathbb{N}) \times \mathbb{M}[\Gamma^* \times \mathbb{N}]$. Given a configuration $\langle g, (w, i), \mathbf{m} \rangle$, we call g the *(global) state*, (w, i) the *local configuration* of the *active thread*, and $\mathbf{m}$ the multiset of the *local configurations* of the *inactive threads*. The initial configuration of $\mathcal{A}$ is $\langle g_0, (\gamma_0, 0), \emptyset \rangle$. For a configuration c of $\mathcal{A}$, we will sometimes write $c.g$ for the state of c and $c.\mathbf{m}$ for the multiset of threads of c (both active and inactive). The *size* of a configuration $c = \langle g, (w, i), \mathbf{m} \rangle$ is defined as $|c| = |w| + \sum_{(w',j) \in \mathbf{m}} |w'|$.

For $i \in \mathbb{N}$ we define the relation $\Rightarrow_i = \rightarrow_i \cup \mapsto_i$ on configurations of $\mathcal{A}$, where $\rightarrow_i$ and $\mapsto_i$ are defined as follows:

- $\langle g, (\gamma.w, i), \mathbf{m} \rangle \rightarrow_i \langle g', (w'.w, i), \mathbf{m}' \rangle$ for all $w \in \Gamma^*$ iff (1) there is a rule $g|\gamma \hookrightarrow g'|w' \in \Delta$ and $\mathbf{m}' = \mathbf{m}$ or (2) there is a rule $g|\gamma \hookrightarrow g'|w' \triangleright \gamma' \in \Delta$ and $\mathbf{m}' = \mathbf{m} \oplus [\![(\gamma', 0)]\!]$.

- $\langle g, (w, i), \mathbf{m} \oplus [\![(w', j)]\!] \rangle \mapsto_i \langle g, (w', j), \mathbf{m} \oplus [\![(w, i+1)]\!] \rangle$ for all $j \in \mathbb{N}$, $g \in G$, $\mathbf{m} \in \mathbb{M}[\Gamma^* \times \mathbb{N}]$, and $w, w' \in \Gamma^*$.

For $b \in \mathbb{N}$ we define the relation $\Rightarrow_{\leq b} := \bigcup_{i=0}^{b} \Rightarrow_i$. We use $\Rightarrow_i^*$ and $\Rightarrow_{\leq b}^*$ to denote the reflexive, transitive closure of $\Rightarrow_i$ and $\Rightarrow_{\leq b}$, respectively.

Given $K \in \mathbb{N}$, a state g of $\mathcal{A}$ is *K-bounded reachable* iff $\langle g_0, (\gamma_0, 0), \emptyset \rangle \Rightarrow_{\leq K}^* \langle g, (w, i), \mathbf{m} \rangle$ for some $(w, i) \in \Gamma^* \times \{0, \ldots, K\}$ and $\mathbf{m} \in \mathbb{M}[\Gamma^* \times \{0, \ldots, K+1\}]$.

Intuitively, a local configuration (w, i) describes a pushdown thread with stack content w that has already performed i context switches. The relation $\rightarrow_i$ corresponds to applying the two kinds of transition rules at i context switches. Both of them define pushdown transitions, which the active thread can perform. Type (2) also spawns a new inactive pushdown thread with 0 context switches, whose initial stack content consists of a single specified symbol. For each $i \in \mathbb{N}$, the relation $\mapsto_i$ corresponds to switching out the active thread and raising its number of context switches from i to $i+1$, while also switching in a previously inactive thread.

3 Results

Our main result concerns the following problem, defined for a fixed K:

K-Bounded State Reachability for DCPS.(SRP[K])	
Input:	A DCPS $\mathcal{A}$ and a global state g.
Task:	Is g K-bounded reachable in $\mathcal{A}$?

This corresponds to asking whether the global state g is reachable if each thread can perform at most K context switches.

Theorem 1 (Main Result)**.** *For each $K \geq 1$, the problem* SRP[K] *is* 2EXPSPACE*-complete.*

The fact that SRP[K] is in 2EXPSPACE for any fixed K follows from the results of Atig et al. [1]. They use a slightly different variant of DCPS. However, it is possible to show a reduction from SRP[K] for our variant to SRP[$K+2$] for theirs.

Our main contribution is to show 2EXPSPACE-hardness for SRP[1]. One may also adapt the results of Atig et al. to the problem where K is part of the input (encoded in unary), to derive an EXPSPACE lower bound and a 2EXPSPACE upper bound. Our result immediately implies 2EXPSPACE-hardness for this problem as well.

The proof involves a series of reductions. We start with the halting problem for triple-exponentially bounded counter programs and reduce it to the halting problem for recursive programs with Petri net counters (RNP). This involves an adapted version of the Lipton construction [11], as presented in [5]. Like in the Lipton construction one can then reduce to Petri net coverability from there. However, in our case the resulting Petri net would be of exponential size. Thus, we introduce *transducer-defined Petri nets* (TDPN), a succinct representation for Petri nets and reduce to their coverability problem. We then show that a TDPN can be simulated by a DCPS using only a single context switch per thread, completing the reduction to SRP[1]. This is achieved by having each thread of the DCPS represent a single token on a Petri net place, with the unique address of the place as its stack content.

Closer inspection of this proof reveals that our lower bound holds already for DCPS where the stack of each thread is bounded by a linear function of the size of the DCPS. Thus, as a corollary, we get 2EXPSPACE-hardness for a related model in which each thread is a *Boolean program*, i.e., where each thread has its stack bounded by a constant but has a polynomial number (in the size of $|G| + |\Gamma| + |\Delta|$) of local Boolean variables. This closes the gap from [7] as well as other similar models studied in the literature [2, 8, 4].

References

[1] Mohamed Faouzi Atig, Ahmed Bouajjani, and Shaz Qadeer. Context-bounded analysis for concurrent programs with dynamic creation of threads. In *Tools and Algorithms for the Construction and Analysis of Systems, 15th International Conference, TACAS 2009, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2009, York, UK, March 22-29, 2009. Proceedings*, pages 107–123, 2009.

[2] Byron Cook, Daniel Kroening, and Natasha Sharygina. Verification of Boolean programs with unbounded thread creation. *Theoretical Computer Science*, 388(1-3):227–242, 2007.

[3] Stéphane Demri, Diego Figueira, and M. Praveen. Reasoning about data repetitions with counter systems. In *28th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2013, New Orleans, LA, USA, June 25-28, 2013*, pages 33–42, 2013.

[4] Emanuele D'Osualdo, Jonathan Kochems, and C.-H. Luke Ong. Automatic verification of Erlang-style concurrency. In *Static Analysis - 20th International Symposium, SAS 2013, Seattle, WA, USA, June 20-22, 2013, Proceedings*, volume 7935 of *Lecture Notes in Computer Science*, pages 454–476. Springer, 2013.

[5] Javier Esparza. Decidability and complexity of Petri net problems – an introduction. In G. Rozenberg and W. Reisig, editors, *Lectures on Petri Nets I: Basic Models. Advances in Petri Nets*, number 1491 in Lecture Notes in Computer Science, pages 374–428, 1998.

[6] Pierre Ganty and Rupak Majumdar. Algorithmic verification of asynchronous programs. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 34(1):6, 2012.

[7] Alexander Kaiser, Daniel Kroening, and Thomas Wahl. Dynamic cutoff detection in parameterized concurrent programs. In *22nd International Conference on Computer Aided Verification, CAV 2010, Edinburgh, UK, July 15-19, 2010, Proceedings*, pages 645–659. Springer, 2010.

[8] Jonathan Kochems. *Verification of asynchronous concurrency and the shaped stack constraint*. PhD thesis, University of Oxford, UK, 2014.

[9] Salvatore La Torre, Parthasarathy Madhusudan, and Gennaro Parlato. The language theory of bounded context-switching. In *LATIN 2010: Theoretical Informatics, 9th Latin American Symposium, Oaxaca, Mexico, April 19-23, 2010, Proceedings*, volume 6034 of *Lecture Notes in Computer Science*, pages 96–107. Springer, 2010.

[10] Akash Lal and Thomas W. Reps. Reducing concurrent analysis under a context bound to sequential analysis. *Formal Methods in System Design*, 35(1):73–97, 2009.

[11] Richard Lipton. The reachability problem is exponential-space hard. *Yale University, Department of Computer Science, Report*, 62, 1976.

[12] Madanlal Musuvathi and Shaz Qadeer. Iterative context bounding for systematic testing of multithreaded programs. In *Proceedings of the ACM SIGPLAN 2007 Conference on Programming Language Design and Implementation, PLDI 2007, San Diego, CA, USA, June 10-13, 2007*, pages 446–455. ACM, 2007.

[13] Shaz Qadeer and Jakob Rehof. Context-bounded model checking of concurrent software. In *Tools and Algorithms for the Construction and Analysis of Systems, 11th International Conference, TACAS 2005, Held as Part of the Joint European Conferences on Theory and*

Practice of Software, ETAPS 2005, Edinburgh, UK, April 4-8, 2005, Proceedings, volume 3440 of *Lecture Notes in Computer Science*, pages 93–107. Springer, 2005.