

Verifying Parametric Thread Creation

Igor Walukiewicz

CNRS, LaBRI, University of Bordeaux, Bordeaux, France

Abstract. Automatic verification of concurrent systems is an active area of research since at least a quarter of a century. We focus here on analyses of systems designed to operate with an arbitrary number of processes. German and Sistla, already in 1992, initiated in depth investigation of this problem for finite state systems. For infinite state systems, like pushdown systems, extra care is needed to avoid undecidability, as reachability is undecidable even for two identical pushdown processes communicating via single variable. Kahlon and Gupta in 2006 have proposed to use parametrization as means of bypassing this undecidability barrier. Indeed when instead of two pushdown processes we consider some unspecified number of them, the reachability problem becomes decidable. This idea of parametrization as an abstraction has been pursued further by Hague, who in 2011 has shown that the problem is still decidable when one of the pushdown processes is made different from the others: there is one leader process and many contributor processes. We discuss how the idea of parametrization as an abstraction leads to decidability, and in some cases even efficient algorithms, for verification of systems which combine recursion with dynamic thread creation.

1 An Overview

We consider recursive programs with thread creation. A thread can be abstracted as a pushdown process. Communication between threads is via global variables as well as via local variables that are shared between a thread and its sub-threads. This setting is an abstraction of a situation found today in many programming languages such as Java, Scala, or Erlang.

While this setting can model many phenomena in programming languages, it is not adapted to automatic verification. Reachability is not decidable even for the case when there are two threads communicating over a 2-bit shared variable. In absence of global variables, reachability becomes undecidable already for two pushdown threads if a rendez-vous primitive is available [19]. A similar result holds if finitely many locks are allowed [11].

We obtain a decidable setting by relaxing the semantics of thread creation operation. Instead of creating one thread the operation creates some unspecified number of threads. The general idea goes back to Kahlon, who observed that various verification problems become decidable for multi-pushdown systems that are *parametric* [10], i.e., systems consisting of an arbitrary number of indistinguishable pushdown threads. Later, Hague extended this result by showing that

an extra designated leader thread can be added without sacrificing decidability [9]. All threads communicate here over a shared, bounded register *without* locking. It is crucial for decidability that only one thread has an identity, and that the operations on the shared variable do not allow to elect a second leader.

The setting of Hague has attracted some attention in recent years. Esparza et al. established the complexity of deciding reachability in that model [7]. La Torre et al. generalized these results to hierarchically nested models for a fixed nesting depth [14]. Durand-Gasselin et al. [6] have shown decidability of the liveness problem for this model. It turns out that the problem has a surprisingly low complexity, namely it is PSPACE-complete [8]. Another problem that has been considered is universal reachability: this is the question of deciding if on every maximal execution trace of the system, the leader reaches some designated state. In terms of temporal logics, reachability is about EF properties, while universal reachability is about AF properties. While still decidable, this problem has very different nature and it turns out to be coNEXPTIME-complete [8]. Indeed, generalizing this result we obtain that all stuttering LTL properties of the leader process can be decided in coNEXPTIME.

The results above concern the case with one leader process that issues one thread creation operation resulting in some number of sub-processes who do not create any new sub-processes. It turns out that we can go even further and have a decidable model for recursive programs with parametric thread creation [18]. Reachability is decidable for a very general class of processes. Every sub-process can maintain a local pushdown store, spawn new sub-processes, and communicate over global variables, as well as via local variables with its sub-processes and with its parent. As in [7, 9, 14], all variables have bounded domains and no locks are allowed.

The algorithm for deciding reachability in this expressive model relies on well-quasi-orders, so its complexity is very high. Yet, there are simpler instances where we know algorithms of a reasonable complexity [18]. As one such instance, we consider the situation where communication between sub-processes is through global variables only. We show that reachability for this model can be effectively reduced to reachability in the model of Hague [7, 9], giving us a precise characterization of the complexity for pushdown threads as PSPACE. As another instance, we consider a parametric variant of *generalized futures* where spawned sub-processes may not only return a single result but create a stream of answers. For that model, we obtain complexities between NP and DEXPTIME. This opens the venue to apply e.g. SAT-solving to check safety properties of such programs.

2 Related Work

There are other approaches than parametrization to get a decidable model of recursive programs with thread creation.

One approach is to consider systems with locks. As we have mentioned, the model with locks is undecidable even if there are no shared variables, no rendez-vous, or other means of communication between processes. Interestingly,

decidability is regained if locking is performed in a disciplined way. This is, e.g., the case for nested [11] and contextual locking [5]. These decidability results have been extended to dynamic pushdown networks as introduced by Bouajjani et al. [4]. This model combines pushdown threads with dynamic thread creation by means of a `spawn` operation, while it ignores any exchange of data between threads. Indeed, reachability of dedicated states or even regular sets of configurations stays decidable in this model, if finitely many global locks together with nested locking [15, 17] or contextual locking [16] are allowed. Such regular sets allow, e.g., to describe undesirable situations such as concurrent execution of conflicting operations.

Another approach is to bound the number of switches of execution contexts. A simple definition of an execution context is a part of an execution when only one process reads from its stack. A context switch is when some other process starts reading from its stack. So the reachability problem now asks for an execution with a given fixed number of context switches. Many decidability results have been established in the last decade for more and more refined notions of context switching [1–3, 12, 13]. In [1, 3], dynamic thread creation is allowed.

References

1. Atig, M.F., Bouajjani, A., Qadeer, S.: Context-bounded analysis for concurrent programs with dynamic creation of threads. *Logical Meth. Comput. Sci.* **7**(4), 1–48 (2011)
2. Bollig, B., Gastin, P., Schubert, J.: Parameterized verification of communicating automata under context bounds. In: Ouaknine, J., Potapov, I., Worrell, J. (eds.) *RP 2014*. LNCS, vol. 8762, pp. 45–57. Springer, Heidelberg (2014). doi:[10.1007/978-3-319-11439-2_4](https://doi.org/10.1007/978-3-319-11439-2_4)
3. Bouajjani, A., Esparza, J., Schwoon, S., Strejcek, J.: Reachability analysis of multithreaded software with asynchronous communication. In: Sarukkai, S., Sen, S. (eds.) *FSTTCS 2005*. LNCS, vol. 3821, pp. 348–359. Springer, Heidelberg (2005). doi:[10.1007/11590156_28](https://doi.org/10.1007/11590156_28)
4. Bouajjani, A., Müller-Olm, M., Touili, T.: Regular symbolic analysis of dynamic networks of pushdown systems. In: Abadi, M., Alfaro, L. (eds.) *CONCUR 2005*. LNCS, vol. 3653, pp. 473–487. Springer, Heidelberg (2005). doi:[10.1007/11539452_36](https://doi.org/10.1007/11539452_36)
5. Chadha, R., Madhusudan, P., Viswanathan, M.: Reachability under contextual locking. In: Flanagan, C., König, B. (eds.) *TACAS 2012*. LNCS, vol. 7214, pp. 437–450. Springer, Heidelberg (2012)
6. Durand-Gasselín, A., Esparza, J., Ganty, P., Majumdar, R.: Model checking parameterized asynchronous shared-memory systems. In: Kroening, D., Păsăreanu, C.S. (eds.) *CAV 2015*. LNCS, vol. 9206, pp. 67–84. Springer, Heidelberg (2015). doi:[10.1007/978-3-319-21690-4_5](https://doi.org/10.1007/978-3-319-21690-4_5)
7. Esparza, J., Ganty, P., Majumdar, R.: Parameterized verification of asynchronous shared-memory systems. *J. ACM* **63**(1), 10 (2016)
8. Fortin, M., Muscholl, A., Walukiewicz, I.: On parametrized verification of asynchronous, shared-memory pushdown systems. *CoRR*, abs/1606.08707 (2016)

9. Hague, M.: Parameterised pushdown systems with non-atomic writes. In: Chakraborty, S., Kumar, A. (eds.) IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 12–14, 2011, Mumbai, India, vol. 13 of LIPIcs, pp. 457–468. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, December 2011
10. Kahlon, V.: Parameterization as abstraction: a tractable approach to the dataflow analysis of concurrent programs. In: Proceedings of the Twenty-Third Annual IEEE Symposium on Logic in Computer Science, LICS 2008, 24–27, Pittsburgh, PA, USA, pp. 181–192. IEEE Computer Society, June 2008
11. Kahlon, V., Ivančić, F., Gupta, A.: Reasoning about threads communicating via locks. In: Etessami, K., Rajamani, S.K. (eds.) CAV 2005. LNCS, vol. 3576, pp. 505–518. Springer, Heidelberg (2005). doi:[10.1007/11513988_49](https://doi.org/10.1007/11513988_49)
12. La Torre, S., Madhusudan, P., Parlato, G.: Model-checking parameterized concurrent programs using linear interfaces. In: Touili, T., Cook, B., Jackson, P. (eds.) CAV 2010. LNCS, vol. 6174, pp. 629–644. Springer, Heidelberg (2010). doi:[10.1007/978-3-642-14295-6_54](https://doi.org/10.1007/978-3-642-14295-6_54)
13. La Torre, S., Madhusudan, P., Parlato, G.: Sequentializing parameterized programs. In: FIT 2012, EPTCS, vol. 87, pp. 34–47 (2012)
14. La Torre, S., Muscholl, A., Walukiewicz, I.: Safety of parametrized asynchronous shared-memory systems is almost always decidable. In: Aceto, L., de Frutos-Escrig, D. (eds.) 26th International Conference on Concurrency Theory, CONCUR, LIPIcs, Madrid, Spain, September 1.4, vol. 42, pp. 72–84. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2015)
15. Lammich, P., Müller-Olm, M.: Conflict analysis of programs with procedures, dynamic thread creation, and monitors. In: Alpuente, M., Vidal, G. (eds.) SAS 2008. LNCS, vol. 5079, pp. 205–220. Springer, Heidelberg (2008)
16. Lammich, P., Müller-Olm, M., Seidl, H., Wenner, A.: Contextual locking for dynamic pushdown networks. In: Logozzo, F., Fähndrich, M. (eds.) SAS 2013. LNCS, vol. 7935, pp. 477–498. Springer, Heidelberg (2013). doi:[10.1007/978-3-642-38856-9_25](https://doi.org/10.1007/978-3-642-38856-9_25)
17. Lammich, P., Müller-Olm, M., Wenner, A.: Predecessor sets of dynamic pushdown networks with tree-regular constraints. In: Bouajjani, A., Maler, O. (eds.) CAV 2009. LNCS, vol. 5643, pp. 525–539. Springer, Heidelberg (2009). doi:[10.1007/978-3-642-02658-4_39](https://doi.org/10.1007/978-3-642-02658-4_39)
18. Muscholl, A., Seidl, H., Walukiewicz, I.: Reachability for dynamic parametric processes. CoRR, abs/1609.05385 (2016)
19. Ramalingam, G.: Context-sensitive synchronization-sensitive analysis is undecidable. ACM Trans. Program. Lang. Syst. **22**(2), 416–430 (2000)

SOFSEM 2017: Theory and Practice of Computer
Science

43rd International Conference on Current Trends in
Theory and Practice of Computer Science, Limerick,
Ireland, January 16-20, 2017, Proceedings

Steffen, B.; Baier, C.; van den Brand, M.; Eder, J.;
Hinchey, M.; Margaria, T. (Eds.)

2017, XVIII, 526 p. 109 illus., Softcover

ISBN: 978-3-319-51962-3