

Reachability Analysis of Communicating Networks with Pushdowns

Alexander Heußner, Jérôme Leroux, Anca Muscholl, Grégoire Sutre

LaBRI, Université Bordeaux – France
ANR AVeriSS

The automatic verification of today’s ubiquitous distributed applications demands for approaches that embrace different means of communication and synchronization. In the following, we will focus on models of *asynchronously* communicating *recursive* programs by introducing the class of *queueing concurrent processes* (QCP) that combines various local automaton models with asynchronous peer-to-peer communication over fifo channels that are reliable, point-to-point, and unbounded.

Although QCP have good expressive power, their reachability — which is the most fundamental question regarding safety verification — is undecidable in general. This is due to their subsumption of communicating finite-state automata, whose reachability is known to be undecidable [BZ83]. Further, adding pushdowns to a finite-state QCP yields an additional source of undecidability owing to pushdown automata synchronization. One of our main motivations is to separate these two sources of undecidability by considering behavioral restrictions that, at the one hand, render reachability decidable, and, on the other hand, conserve the simplicity and expressiveness of the model.

Our point of departure is the work of La Torre et al. [LMP08] that allowed to derive decidability results for directed forest architectures of “*well-queueing*” recursive QCP (RQCP). Well-queueing demands that a pushdown automaton can only dequeue/receive from a fifo queue if its stack is empty (but poses no restriction on enqueueing/sending). This corresponds to an event-based programming paradigm: tasks are executed by threads without interruption; when the local computation is finished (i.e., the stack is empty), the next task is started. Their work allowed to derive the decidability of reachability for well-queueing RQCP if the underlying network architecture is a *directed forest*. In the decidable case, their paper derived a doubly exponential upper bound by reduction to bounded-phase multi-stack pushdown automata (MSPDS) [LMP07]. Further, they also provided a second approach, inspired by recent work on reachability with bounded contexts [QR05], that allows to decide bounded-context reachability for well-queueing RQCP with a time bounded doubly exponentially in the number of contexts. Again, this is proven by reduction to bounded-phase MSPDS.

We extend the work of La Torre et al. in several directions. First, we add a dual notion to well-queueing: a pushdown process can enqueue (send) messages

only with empty stack (but can dequeue messages without restriction). This corresponds to an interrupt based programming paradigm where a thread can receive messages, e.g., tasks of a higher priority, while executing, but it can only send its result after the (recursive) computation is finished. *Oriented* architectures combine these two dual notions by fixing the behavior of the pushdown processes at each channel’s two endpoints.

Second, we characterize the class of RQCP over oriented architectures that exhibits decidable reachability when restricting the runs to be *eager*. Informally, eagerness demands that the sending of a message is immediately followed by its reception, a notion closely connected to existentially 1-bounded communication [LM04]. Bounded communication for networks of finite automata is known to be decidable [GKM06], and, hence, allows us to investigate the influence of adding pushdowns to a finite-state model.

Main Results. An oriented architecture is called *confluent* if it contains a process that can communicate via the network with two distinct other processes that, with respect to this connection, can both use their pushdown stacks without restriction. We show that reachability of RQCP over eager runs is decidable for oriented architectures iff they are non-confluent. In the latter case, our constructions allow to derive EXPTIME-*completeness*.

Eagerness is a relatively strong requirement, hence, we show how it arises naturally by imposing a semantic restriction on the communication flow: the *mutex* restriction demands that in every reachable configuration there is no more than one non-empty channel per communication cycle. Thus, mutex can be seen as a generalization of the half-duplex restriction investigated in [CF05]. Obviously, polyforest architectures are mutex, and we can directly subsume La Torre et al.’s results. This idea can further be transferred to other classes of—possibly *infinite*—QCP, e.g., based on communicating Petri nets.

Our second main contribution is the extension of the bounded-context result to RQCP over oriented architectures. We provide a direct constructive proof that includes both well-queueing channels and their dual, as well as avoids [LMP08]’s reduction to MSPDS.

Practical Implications. Our approach allows to include both interrupt-driven and event-based threads that communicate under certain conditions.

The focus on unbounded, reliable, asynchronous fifo communication mirrors out interest to verify applications based on an underlying layer of TCP, e.g., client-server or peer-2-peer implementations based on the Berkeley Sockets API [Heu09].

Cyclic or mutual communication is the central property of most practical applications but is most often excluded in formal models in order to avoid undecidability (see the directed tree architectures of [LMP08]). The mutex property together with non-confluency allows us to model important cyclic architectures like (token) rings, or bidirectional communication in hierarchical overlay networks which are on the rise with the current focus on Grid computing (viz. Fig. 1).

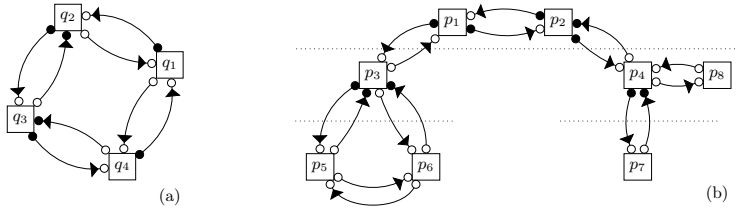


Fig. 1. Non-confluent architectures: (a) a mutex ring (\rightarrow denotes the direction of the channel whose labelling \circ/\bullet represents the restricted/unrestricted usage of the push-down with respect to this channel), and (b) a hierarchical master-worker setting — tree-like architecture with bidirectional channels between master and workers (distribute tasks and collect results while in computation, send result to own master when computation is finished, i.e., stack empty) as well as channels whose both ends are restricted between workers of the same master.

This abstract is based on [HLMS10]. A long version of this paper that includes all proofs that were omitted due to space limitations can be obtained at <http://hal.archives-ouvertes.fr/hal-00443529/>.

References

- [BZ83] D. Brand, P. Zafiropulo. On Communicating Finite-State Machines. *J. ACM*, 30(2):323–342, 1983.
- [CF05] G. Cécé, A. Finkel. Verification of programs with half-duplex communication. *Inf. Comput.*, 202(2):166–190, 2005.
- [GKM06] B. Genest, D. Kuske, A. Muscholl. A Kleene theorem and model checking algorithms for existentially bounded communicating automata. *Inf. Comput.*, 204(6):920–956, 2006.
- [Heu09] A. Heußner. Model extraction for sockets-based distributed programs. Technical report, LaBRI, October 2009.
- [HLMS10] A. Heußner, J. Leroux, A. Muscholl, G. Sutre. Reachability analysis of communicating pushdown systems. in *Proceedings of FOSSACS '10*, 267–281. Springer, 2010.
- [LMP07] S. La Torre, P. Madhusudan, G. Parlato. A robust class of context-sensitive languages. in *Proceedings of LICS '07*, 161–170. IEEE Computer Society, 2007.
- [LMP08] S. La Torre, P. Madhusudan, G. Parlato. Context-bounded analysis of concurrent queue systems. in *Proceedings of TACAS '08*, 299–314. Springer, 2008.
- [LM04] M. Lohrey, A. Muscholl. Bounded MSC communication. *Inf. Comput.*, 189(2):160–181, 2004.
- [QR05] S. Qadeer, J. Rehof. Context-bounded model checking of concurrent software. in *Proceedings of TACAS '05*, 93–107. Springer, 2005.