Higher-Order Model-Checking and Underapproximate Models of Concurrent Recursive Programs

Pushdown automata (PDA) are a good model of recursive programs which also have good algorithmic properties. For example "reachability" for PDA is decidable, allowing one to automatically check whether a program modelled by the PDA is "safe"—i.e. that it cannot "reach an error state".

But what happens if one wants to model a recursive program featuring more than one thread of execution? A nave idea would be to generalise pushdown automata by allowing them multiple stacks—one for each thread. Unfortunately, as is well known, PDAs with more than one stack are as powerful as Turing machines; this precludes the possibility of them having any interesting decidable properties.

One possible approach to this problem is to impose constraints on the ways in which multi-stack PDA can use their stacks. On the plus side these constraints should produce a model for which reachability is decidable. On the down side, the constrained model will in general only be able to "underapproximate" the behaviour of concurrent recursive programs—the program may exhibit behaviours that a constrained multi-stack PDA is unable to replicate. Whilst the resulting techniques may not be able to provide "formal guarantees" about program safety, the idea is that they still "explore a sufficient amount of behaviour" to "stand a good chance" of discovering any bugs that may exist.

An example of such a model is "context-bounded multi-stack PDA" [7]. The PDA performs a "context switch" when it pops from a stack different to that from which it last popped. A "context-bounded multi-stack PDA" is only allowed to perform up to n context switches for some fixed constant n. Several other constraints have also been considered (e.g. [1, 5, 6]).

Let us now switch our attention to a different line of research. "Higher-order model-checking" concerns itself with checking safety properties of "higher-order recursion schemes" (HORS), which are essentially terms in the simply-typed lambda calculus with recursion. In recent years much effort has been invested in developing higher-order model-checkers that perform well in practice. Most of the algorithms are formulated within the framework proposed by Kobayashi [3] in which so-called "intersection types" are used to reason about safety. One advantage of this approach is that if the algorithm determines that a HORS is safe, then it can also output "intersection types" that certify safety. Since such a

certificate can be checked by a trivial algorithm, this provides a way of checking that the implementation of the model-checker really has given a correct answer (and has not erred due to some bug).

Whilst most applications of higher-order model-checking concern the verification of programs that employ higher-order functions, HORS constitute a highly expressive model which one would expect to have much broader applicability. This thesis will investigate using higher-order model-checking to model-check various flavours of (constrained) multi-stack PDA. The following would be a possible line of attack:

1. Order-1 recursion schemes have the same expressive power as (single-stack) PDA. The thesis will define various kinds of "order-1 concurrent" recursion schemes that have the same expressive power as the various kinds of multi-stack PDA.

2. An "intersection type system" should be developed for these concurrent recursion schemes that allows one to reason about their safety in a manner analogous to that for standard order-1 HORS.

3. The model-checking (of safety properties) of order-1 concurrent recursion schemes (in their various flavours) should be shown to be reducible to model-checking HORS (of higher-orders). Moreover a method should be developed for deducing intersection types for a safe order-1 concurrent recursion scheme from the intersection types inferred by a HORS model-checker. This would allow existing higher-order model-checkers to be employed in the verification of multi-stack PDA models.

   (Kobayashi includes a reduction from model-checking a kind of 'bounded context-switching' concurrent HORS to model-checking standard HORS as an example in the appendix of [4].)

4. If time permits, the thesis could investigate generalising "order-1 concurrent recursion schemes" to higher-orders. This could possibly but not necessarily take inspiration from [2] (which proposes a way of doing something analogous for a generalisation of PDA equi-expressive with HORS called "collapsible pushdown automata").

5. Another possible enhancement would be to investigate performing so-called "partial-order reductions" on concurrent recursion schemes and reflecting this in the intersection type system. (Partial-order reduction is a technique for reducing the search space when model-checking concurrent systems without impairing the accuracy of the result).

References

[1] Mohamed Faouzi Atig, Benedikt Bollig, and Peter Habermehl. "Emptiness of multi-pushdown automata is 2ETIME-complete". In: *Developments in Language Theory*. Springer. 2008, pp. 121–133.

[2] Matthew Hague. "Saturation of concurrent collapsible pushdown systems". In: *arXiv preprint arXiv:1310.2631* (2013).

[3] Naoki Kobayashi. "Types and higher-order recursion schemes for verification of higher-order programs". In: *ACM SIGPLAN Notices*. Vol. 44. 1. ACM. 2009, pp. 416–428.

[4] Naoki Kobayashi and Atsushi Igarashi. "Model-checking higher-order programs with recursive types". In: *Programming Languages and Systems*. Springer, 2013, pp. 431–450.

[5] Salvatore La Torre, Parthasarathy Madhusudan, and Gennaro Parlato. "A robust class of context-sensitive languages". In: *Logic in Computer Science, 2007. LICS 2007. 22nd Annual IEEE Symposium on*. IEEE. 2007, pp. 161–170.

[6] Salvatore La Torre and Margherita Napoli. "Reachability of multistack pushdown systems with scope-bounded matching relations". In: *CONCUR 2011–Concurrency Theory*. Springer, 2011, pp. 203–218.

[7] Shaz Qadeer and Jakob Rehof. "Context-bounded model checking of concurrent software". In: *Tools and Algorithms for the Construction and Analysis of Systems*. Springer, 2005, pp. 93–107.