Unbounded data nesting for loops in higher-order programs*

Adriana Baldacchino Andrzej Murawski

May 2026

Programming languages from the ML family enrich purely functional computation with effects, such as the ability to store values of arbitrary types (general store) and to manipulate control flow (continuations and call/cc). This combination of features gives rise to phenomena that often break intuitions from classic imperative programming, leading to unexpected consequences. Consider, for example, the question whether the following term (with a free variable $f : \mathsf{Unit} \to \mathsf{Unit}$) can be placed in a context that drives the computation to reach the call $err()$. Note that, in ML-like languages, $!r$ stands for dereferencing rather than negation.

$$\begin{array}{l}\textbf{let}\, r = \textbf{ref}\, \textbf{true}\, \textbf{in} \\ \quad \textbf{while}\, !r\, \textbf{do}\, (r := \textbf{false}; f(); r := \textbf{true}; f()); \\ \quad err()\end{array}$$

It might seem that, because the assignment $r := \textbf{true}$ always follows $r := \textbf{false}$, the loop is bound to run forever. This intuition is indeed correct in the absence of control effects and references other than those to ground values. However, with general references and call/cc, it is possible to instrument a context in which the above term does call err (by capturing the continuation immediately before the guard is tested and invoking it when r is set to **false**). Our work is dedicated to studying the *contextual reachability* problem sketched above.

We also study *contextual approximation*, which asks given two terms t_1, t_2, whether the observable behaviour of t_1 is an observable behaviour of t_2, within all contexts. This property is used to justify program transformations, as if we can prove a term contextually approximates another, we can replace one with the other, preserving the observable behaviour.

For both these directions, we focus on the addition of loops, as this is a limitation of previous work [2]. We study contextual interactions of programs over finite datatypes with ground-type references, continuations and loops (which we refer to as FOSC + **while**) over contexts with unrestricted storage (referred to as HOSC). We show that with the addition of loops, decidability of reachability is preserved, but contextual approximation, which is decidable in the loop-free case [2], becomes undecidable.

We study these two problems using the operational game models in [4]. In these models, interactions between programs and contexts are represented as interactions between two players, the context O and the program P. Computations are described by alternating traces of calls and returns annotated with freshly generated names, which record the flow of control between the two players. A central notion in these models arising from game semantics is the P-view, which specifies a fragment of the trace that captures all resources currently in scope.

In the loop-free case, P-views are bounded, which allows the traces generated by these models to be captured by weak Nested Data Class Memory Automata (NDCMA) [2], which were introduced in [3]. NDCMA operate over an infinite alphabet, which is structured as an infinitely-branching bounded depth forest. These automata can hence store P-views as bounded-depth branches of data. This, along with decidability results for these automata leads to decidability results for the loop-free case.

*This talk is based upon the paper of the same name appearing at LICS 2026 [1].

With the addition of loops, things become more complicated as P-views are no longer bounded. For example, term $f : \mathsf{Unit} \to \mathsf{Unit} \vdash \textbf{while true do } f() : \mathsf{Unit}$ generates traces of the form

$$\bar{f}((), c_1)\ c_1(())\ \bar{f}((), c_2)\ c_2(())\cdots\bar{f}((), c_n)\ c_n(())$$

whose P-view is the entire trace. To deal with this unboundedness, we introduce *Unbounded Nested Data Class Memory Automata* (UNDCMA), which operate over an infinite alphabet with unbounded branching and unbounded depth. To recover a finite description of transitions, each data value is equipped with a bounded *list memory*, which specifies the data values it can use to determine its transition. This memory consists only of ancestors of a data value and behaves similarly to a stack. When a data value d is seen for the first time, its list memory is determined from its parent $pred(d)$ and its parent's memory $d_1, \ldots, d_k$, and is taken to be a (bounded) suffix of $pred(d), d_1, \ldots, d_k$. This way, the structure mimics that of a scope, where visibility is transitive.

In parallel to the list memory, which restricts the visible part of an unbounded branch, we introduce restrictions to the P-view which we call the *shallow P-view* and *enriched shallow P-view*. The shallow P-view reflects the section of the P-view with actions from completed iterations removed, and the enriched shallow P-view extends the shallow P-view with extra actions that serve as allocation points for names and locations. These views are execution dependent, and hence must not be exposed directly in the trace representation so as not to reveal the locations of loops in the program.

Our main results are:

- FOSC+**while** terms in HOSC contexts correspond directly to deterministic UNDCMA (dUNDCMA).
- Reachability is decidable for dUNDCMA and trace inclusion is undecidable for dUNDCMA.
- Contextual reachability (with respect to HOSC contexts) is decidable for FOSC+while, while contextual approximation is undecidable.

In this talk we explain how we reach this correspondence between game models and automata, focusing on the challenges in reducing the LTS to be representable by our automata.

References

[1] Adriana Baldacchino and Andrzej S. Murawski. Contextual equivalence for state and control via nested data. In *Proceedings of the 41st Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2026, Lisbon, Portugal, July 20-23*, 2026.

[2] Benedict Bunting and Andrzej S. Murawski. Contextual equivalence for state and control via nested data. In Pawel Sobocinski, Ugo Dal Lago, and Javier Esparza, editors, *Proceedings of the 39th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2024, Tallinn, Estonia, July 8-11, 2024*, pages 19:1–19:14. ACM, 2024.

[3] Conrad Cotton-Barratt, Andrzej S. Murawski, and C.-H. Luke Ong. Weak and Nested Class Memory Automata. In *Language and Automata Theory and Applications*, pages 188–199. Springer International Publishing, 2015.

[4] Guilhem Jaber and Andrzej S. Murawski. Complete trace models of state and control. In *Proceedings of ESOP*, pages 348–374. Springer International Publishing, 2021.