Abstract

Our paper introduces a systematic approach to the alignment of names in the static structure of a program, and memory layout and access during its execution. We develop a uniform memory model consisting of frames that instantiate the scopes in the scope graph of a program. This provides a language-independent correspondence between static scopes and run-time memory layout, and between static resolution paths and run-time memory access paths. The approach scales to a range of binding features, supports straightforward type soundness proofs, and provides the basis for a language-independent specification of sound reachability-based garbage collection.

This Coq artifact showcases how our uniform model for memory layout in dynamic semantics provides structure to type soundness proofs. The artifact contains type soundness proofs mechanized in Coq for (supersets of) all languages in the paper. The type soundness proofs rely on a language-independent framework formalizing scope graphs and frame heaps.

1 Scope

The artifact is designed to document and support repeatability of the type soundness proofs in the companion paper [2], using the Coq proof assistant. In particular, the artifact provides a
Scopes Describe Frames (Artifact)

A language-independent framework formalizing scope graphs and frame heaps. The scopes-as-frames correspondence is formalized based on this framework, and gives rise to a suite of helper lemmas that are useful for proving the soundness of languages and their type systems, as well as garbage collection (GC) strategies.

## 2 Content

The artifact package includes:

- Coq libraries formalizing scope graphs, frames, and the scopes-describe-frames correspondence;
- Coq formalizations and proofs of soundness for three example languages: L1; a superset of L2 (differences summarized below); and a superset of the L3 language with class-based inheritance and sub-typing that is briefly described in the companion paper, and covered in more detail in the companion technical report [3];
- instructions for using the artifact and for rebuilding it from scratch, provided as a README file; and
- pretty-printed versions of the Coq proof scripts.

### Differences from paper.

The languages described in our paper are simplified versions of the languages that we used to experiment with the scopes-describe-frames correspondence and its application to type and GC soundness. There are numerous small differences of naming and terminology. In addition, the languages in this artifact differ from the paper in the following ways:

- L1 follows the semantics in the paper.
- L2 and L3 differ from the paper as follows:
  - Functions and function types are n-ary, and argument values are stored in call-frames immediately after they are computed (as opposed to in the big-step derivation tree).
  - The language provides three variants of n-ary let-binding: sequential lets, parallel lets, and recursive lets (following the static semantics given for these in [1, 4]). Recursive lets are restricted to bind values of function type only.
  - The language has boolean expressions and simple if-then-else branching.

## 3 Getting the artifact

The artifact endorsed by the Artifact Evaluation Committee is available free of charge on the Dagstuhl Research Online Publication Server (DROPS). The latest version of our code is available on Github: https://github.com/metaborg/scopes-describe-frames.

## 4 Tested platforms

The artifact is known to type check using Coq 8.5.

## 5 License

Apache 2.0 (http://www.apache.org/licenses/LICENSE-2.0)

## 6 MD5 sum of the artifact

41cd2fd85e8469a2ab693515fe291955
7 Size of the artifact

610KB

Acknowledgments. We thank the anonymous reviewers for their feedback on a previous version of the artifact. This research was partially funded by the NWO VICI Language Designer’s Workbench project (639.023.206). Andrew Tolmach was partly supported by a Digiteo Chair at Laboratoire de Recherche en Informatique, Université Paris-Sud.

References