Title

Modeling a subset of Featherweight Java in ACL2

Background

Featherweight Java or FJ is a simple subset of Java[1]. Its functionalities are limited to the core of OOP, such as variables, field access, method invocation, object creation, and casts. In the programming language research area, using theorem provers for proof of type soundness is not uncommon. Some cases utilize Coq and Isabelle, but a few examples use ACL2, especially for an object-oriented language. Therefore, we will use ACL2 for proving type soundness and choose FJ, a well-known Subset of Java, as the target language.

Technical Approach

We will deal with the subset of FJ instead of the whole FJ because it is still huge to model and prove in ACL2 in a month. Therefore, we model FJ without a subtyping feature. That is, there is no class inheritance, polymorphism, and casts. We refer the subset of featherweight java as SFJ, for short in this report.

Our goals of this projects are follows.

1. Formalizing SFJ in ACL2.

2. Proving its type soundness in ACL2.

Type soundness is a property that assures typed terms will never be evaluated to undefined terms. We are thinking of modeling the expression by "defsum" and the syntax by "defaggregate". The Evaluator and type checker would be modeled as a function.

References

1. Igarashi, A., Pierce, B. C., & Wadler, P. (2001). Featherweight Java: a minimal core calculus for Java and GJ. ACM Transactions on Programming Languages and Systems (TOPLAS), 23(3), 396-450.

2. Swords, S., & Cook, W. R. (2006, August). Soundness of the simply typed lambda calculus in ACL2. In Proceedings of the sixth international workshop on the ACL2 theorem prover and its applications (pp. 35-39).

Syntax:

L ::= class C extends C {$\overline{C}$ $\overline{f}$; K $\overline{M}$}

K ::= C($\overline{C}$ $\overline{f}$){super($\overline{f}$); this.$\overline{f}$=$\overline{f}$;}

M ::= C m($\overline{C}$ $\overline{x}$){ return e; }

e ::= x | e.f | e.m($\overline{e}$) | new C($\overline{e}$) | (C)e

図 1: Syntax of FJ