Reasoning in Abella about Structural Operational Semantics Specifications

Andrew Gacek\textsuperscript{1}  Dale Miller\textsuperscript{2}  Gopalan Nadathur\textsuperscript{1}

\textsuperscript{1}Department of Computer Science and Engineering
University of Minnesota

\textsuperscript{2}INRIA Saclay - Île-de-France
& LIX/École polytechnique

LFMTP’08
June 23, 2008
$F_{\leq}$

$\lambda$-calculus

sequent calculus

specification logic

meta-logic

$\pi$-calculus
Two-level logic approach

Originally advocated by McDowell & Miller [ToCL02]

Benefits

▶ clean separation between specification and reasoning
▶ features of each logic can be tailored to needs (e.g., executable vs rich)
▶ allows formal meta-theorems about specification logic
▶ allows for different specification logics
Specification logic: $hH^2$

Second-order hereditary Harrop formulas ($hH^2$) provide a simple and expressive logic for specification

$$\forall m, n, a, b[\text{of } m (\text{arr } a b) \land \text{of } n a \supset \text{of } (\text{app } m n) b]$$

$$\forall r, a, b[\forall x[\text{of } x a \supset \text{of } (r x) b] \supset \text{of } (\text{abs } a r) (\text{arr } a b)]$$

This logic is a subset of the logic behind $\lambda$Prolog

$$\text{of } (\text{app } M N) B :$$
$$\qquad : \text{of } M (\text{arr } A B), \text{, of } N A.$$

$$\text{of } (\text{abs } A R) (\text{arr } A B) :$$
$$\qquad : \text{pi } x \backslash \text{of } x A \Rightarrow \text{of } (R x) B.$$

In fact, an efficient implementation of $\lambda$Prolog also exists:

http://teyjus.cs.umn.edu/
Meta-logic: \( G \)

Features

- \( \lambda \)-tree syntax
- \( \nabla \)-quantifier for generic judgments
- induction over natural numbers
- recursive definitions
∇ quantifier: generic judgments

Miller & Tiu “Generic Judgments” [LICS03, ToCL05]
Tiu “LGω” [LFMTP06]

∇ₓ.F means F has a generic proof—one which depends on the freshness, but not the form of x

∀ₓ.F ⊃ ∇ₓ.F
∇ₓ.F ⊄ ∀ₓ.F

∇ₓ.∇y.F ≡ ∇y.∇x.F

∇ₓ.F ≡ F if x does not appear in F

These structural rules allow a treatment of ∇ based on nominal constants which make quantification implicit
Representation technique

**Technique**
We represent bound variables with $\lambda$-terms and “free variables” with nominal constants ($\nabla$)

**Benefits**
- $\alpha$-equivalence and substitution built-in for bound variables
- equivariance built-in for free variables
Role of definitions in $\mathcal{G}$

Logically, definitions for atomic predicates are used to introduce atomic judgments on the left and right sides of a sequent

- on the right, this corresponds to backchaining
- on the left, this corresponds to case-analysis

\[
\text{member } A \ ((A :: L) \triangleq \top)
\]
\[
\text{member } A \ ((B :: L) \triangleq \text{member } A L)
\]

For us, definitions serve two purposes

- encode the semantics of the specification logic
- encode properties of specifications which are relevant to reasoning
Encoding \( hH^2 \) in \( G \)

\( seq_N \ L \ G \) encodes that \( G \) is provable in \( hH^2 \) from the hypotheses \( L \) with at most height \( N \)

\[
\begin{align*}
seq_N \ L \langle A \rangle & \triangleq \text{member } A \ L \\
seq_{(s \ N)} \ L \ (B \land C) & \triangleq seq_N \ L \ B \land seq_N \ L \ C \\
seq_{(s \ N)} \ L \ (A \supset B) & \triangleq seq_N \ (A :: L) \ B \\
seq_{(s \ N)} \ L \ (\forall B) & \triangleq \forall x. seq_N \ L \ (B \ x) \\
seq_{(s \ N)} \ L \langle A \rangle & \triangleq \exists b. prog \ A \ b \land seq_N \ L \ b
\end{align*}
\]

Example \( prog \) clause:

\[
prog \ (of \ (app \ M \ N) \ B) \ (\langle of \ M \ (arr \ A \ B) \rangle \land \langle of \ N \ A \rangle) \triangleq \top
\]
Theorems about typing

Notation: $L \vdash G$ abbreviates $\exists n. \text{nat } n \land \text{seq}_n L \ G$

When $L$ is nil, we write simply $\vdash G$

Type substitution theorem:

$$\forall L, t_1, t_2, a, b. \land x. ((\langle \text{of } x \ a \rangle :: L) \vdash \langle \text{of } (t_1 \ x) \ b \rangle) \land (L \vdash \langle \text{of } t_2 \ a \rangle) \supset (L \vdash \langle \text{of } (t_1 \ t_2) \ b \rangle)$$

Context permutation lemma:

$$\forall L_1, L_2, t, b. (L_1 \vdash \langle \text{of } t \ c \rangle) \land \text{permute } L_1 \ L_2 \supset (L_2 \vdash \langle \text{of } t \ c \rangle)$$
Theorems about \textit{seq}

Contexts admit weakening, contraction, and permutation

\[
\text{subset } L_1 \ L_2 \triangleq \forall X. \text{member } X \ L_1 \supset \text{member } X \ L_2
\]

\[
\forall L_1, L_2, G. \ (L_1 \vdash G) \land \text{subset } L_1 \ L_2 \supset (L_2 \vdash G)
\]

Instantiation for specification logic \(\forall\) quantifier

\[
\forall L, G. \ (\nabla x. (L \ x) \vdash (G \ x)) \supset \forall T. (L \ T) \vdash (G \ T)
\]

Discharging assumptions (cut admissibility)

\[
\forall L, A, G. \ (A :: L \vdash G) \land (L \vdash \langle A \rangle) \supset (L \vdash G)
\]
Implicit properties of specifications

∀t, a₁, a₂. (\models \langle of t a_1 \rangle) \land (\models \langle of t a_2 \rangle) ⊃ a_1 = a_2

∀L, t, a₁, a₂. (L \models \langle of t a_1 \rangle) \land (L \models \langle of t a_2 \rangle) ⊃ a_1 = a_2

∀L, t, a₁, a₂. cntx L \land (L \models \langle of t a_1 \rangle) \land (L \models \langle of t a_2 \rangle) ⊃ a_1 = a_2

cntx L should enforce

▷ L = (of x₁ a₁) :: (of x₂ a₂) :: ... :: (of x_n a_n) :: nil

▷ Each xᵢ is atomic

▷ Each xᵢ is unique
Extended form of definitions

Definitional clauses now take the form

\[ \forall \vec{x}. (\nabla \vec{z}. H) \triangleq B \]

That is, we permit \( \nabla \) quantification over the head

Examples

\((\nabla x. \text{name } x) \triangleq \top\)

\(\forall E. (\nabla x. \text{fresh } x E) \triangleq \top\)

\(\forall E, V. (\nabla x. \text{subst } (E x) x V (E V)) \triangleq \top\)

\(\text{cntx } \text{nil} \triangleq \top\)

\(\forall L, A. (\nabla x. \text{cntx } ((\text{of } x A) :: L)) \triangleq \text{cntx } L\)
Abella

Abella (Gacek 2008) is an interactive, tactics-based implementation of $G$ which focuses on the two-level logic approach and hides most of the supporting machinery.

Proofs done with Abella

- determinacy and type preservation of various evaluation strategies
- POPLmark 1a, 2a
- cut admissibility for a sequent calculus
- Church-Rosser property for $\lambda$-calculus
- Tait-style weak normalizability proof

http://abella.cs.umn.edu/
Key parts of weak normalizability proof

The logical relation

\[
reduce M \, i \triangleq (\vdash \langle \text{of } M \, i \rangle) \land \text{halts } M
\]

\[
reduce M \, (\text{arr } A \, B) \triangleq (\vdash \langle \text{of } M \, (\text{arr } A \, B) \rangle) \land \text{halts } M \land \\
\forall N. (reduce N \, A \supset reduce \, (\text{app } M \, N) \, B)
\]

Substitution and freshness results

\[
\text{subst } \text{nil} \, M \, M \triangleq \top
\]

\[
(\forall \, x. \text{subst } ((\text{of } x \, A) :: L) \, (R \, x) \, M) \triangleq \\
\exists V. \, reduce \, V \, A \land (\vdash \langle \text{value } V \rangle) \land \text{subst } L \, (R \, V) \, M
\]
Related Work

Locally nameless representation
A first-order representation with de Bruijn indices for bound variables and names for free variables [Aydemir et. al. PoPL08]

Nominal logic approach
A formalization of bound and free variable names in an existing theorem prover (Isabelle/HOL) [Urban and Tasson CADE04]

Twelf
An expressive specification logic (LF) with a relatively weak meta-logic ($\mathcal{M}_2^+$) [Schürrmann and Pfenning CADE98]
Conclusions

Benefits of a two-level logic approach

▶ clean separation between specification and reasoning
▶ features of each logic can be tailored to needs (e.g., executable vs rich)
▶ allows formal meta-theorems about specification logic
▶ allows for different specification logics

Moreover, we have found this approach very practical

Future work

▶ richer (co)induction in the meta-logic
▶ alternate specification logics, e.g., linear
▶ proof search, focusing, automation
▶ encoding other parts of the specification logic, e.g., types