

Reasoning in Abella about Structural Operational Semantics Specifications

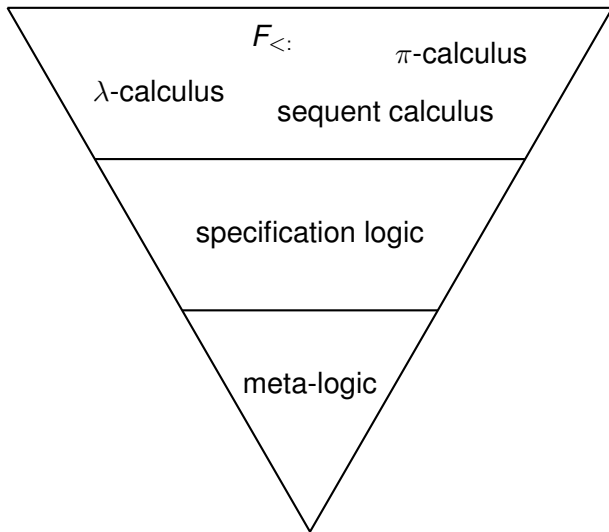
Andrew Gacek¹ Dale Miller² Gopalan Nadathur¹

¹Department of Computer Science and Engineering
University of Minnesota

²INRIA Saclay - Île-de-France
& LIX/École polytechnique

LFMTP'08
June 23, 2008

Preview



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

$$\forall m, n, a, b [of\ m\ (arr\ a\ b) \wedge of\ n\ a \supset of\ (app\ m\ n)\ b]$$
$$\forall r, a, b [\forall x [of\ x\ a \supset of\ (r\ x)\ b] \supset of\ (abs\ a\ r)\ (arr\ a\ b)]$$

This logic is a subset of the logic behind λ Prolog

```
of (app M N) B :-  
  of M (arr A B), of N A.  
of (abs A R) (arr A B) :-  
  pi x \ of x A => of (R x) B.
```

In fact, an efficient implementation of λ Prolog also exists:

<http://teyjus.cs.umn.edu/>

Meta-logic: \mathcal{G}

Features

- ▶ λ -tree syntax
- ▶ ∇ -quantifier for generic judgments
- ▶ induction over natural numbers
- ▶ recursive definitions

∇ quantifier: generic judgments

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

Tiu “ LG^ω ” [LFMTP06]

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

$$\forall x.F \supset \nabla x.F \qquad \nabla x.F \not\supset \forall x.F$$

$$\nabla x.\nabla y.F \equiv \nabla y.\nabla x.F$$

$$\nabla x.F \equiv F \quad \text{if } x \text{ does not appear in } F$$

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

Representation technique

Technique

We represent bound variables with λ -terms and “free variables” with nominal constants (∇)

Benefits

- ▶ α -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 \mathcal{G}

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

$$seq_N L \langle A \rangle \quad \triangleq \text{member } A L$$

$$seq_{(s N)} L (B \wedge C) \triangleq seq_N L B \wedge seq_N L C$$

$$seq_{(s N)} L (A \supset B) \triangleq seq_N (A :: L) B$$

$$seq_{(s N)} L (\forall B) \quad \triangleq \forall x. seq_N L (B x)$$

$$seq_{(s N)} L \langle A \rangle \quad \triangleq \exists b. prog A b \wedge seq_N L b$$

Example *prog* clause:

$$prog (of (app M N) B) (\langle of M (arr A B) \rangle \wedge \langle of N A \rangle) \triangleq \top$$

Theorems about typing

Notation: $L \Vdash G$ abbreviates $\exists n. \text{nat } n \wedge \text{seq}_n L G$
When L is *nil*, we write simply $\Vdash G$

Type substitution theorem:

$\forall L, t_1, t_2, a, b. \nabla x.$

$$\begin{aligned} &(((\text{of } x \ a) :: L) \Vdash \langle \text{of } (t_1 \ x) \ b \rangle) \wedge (L \Vdash \langle \text{of } t_2 \ a \rangle) \supset \\ & \qquad \qquad \qquad (L \Vdash \langle \text{of } (t_1 \ t_2) \ b \rangle) \end{aligned}$$

Context permutation lemma:

$$\forall L_1, L_2, t, b. (L_1 \Vdash \langle \text{of } t \ c \rangle) \wedge \text{permute } L_1 \ L_2 \supset (L_2 \Vdash \langle \text{of } t \ c \rangle)$$

Theorems about *seq*

Contexts admit weakening, contraction, and permutation

$$\textit{subset } L_1 L_2 \triangleq \forall X. \textit{member } X L_1 \supset \textit{member } X L_2$$

$$\forall L_1, L_2, G. (L_1 \Vdash G) \wedge \textit{subset } L_1 L_2 \supset (L_2 \Vdash G)$$

Instantiation for specification logic \forall quantifier

$$\forall L, G. (\forall x. (L x) \Vdash (G x)) \supset \forall T. (L T) \Vdash (G T)$$

Discharging assumptions (cut admissibility)

$$\forall L, A, G. (A :: L \Vdash G) \wedge (L \Vdash \langle A \rangle) \supset (L \Vdash G)$$

Implicit properties of specifications

$$\forall t, a_1, a_2. (\Vdash \langle \text{of } t \ a_1 \rangle) \wedge (\Vdash \langle \text{of } t \ a_2 \rangle) \supset a_1 = a_2$$

$$\forall L, t, a_1, a_2. (L \Vdash \langle \text{of } t \ a_1 \rangle) \wedge (L \Vdash \langle \text{of } t \ a_2 \rangle) \supset a_1 = a_2$$

$$\forall L, t, a_1, a_2. \text{cntx } L \wedge (L \Vdash \langle \text{of } t \ a_1 \rangle) \wedge (L \Vdash \langle \text{of } t \ a_2 \rangle) \supset a_1 = a_2$$

cntx L should enforce

- ▶ $L = (\text{of } x_1 \ a_1) :: (\text{of } x_2 \ a_2) :: \dots :: (\text{of } x_n \ a_n) :: \text{nil}$
- ▶ Each x_j is atomic
- ▶ Each x_j 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 ∇ quantification over the head

Examples

$$(\nabla x. \textit{name } x) \triangleq \top$$

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

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

$$\textit{ctx nil} \triangleq \top$$

$$\forall L, A. (\nabla x. \textit{ctx } ((\textit{of } x A) :: L)) \triangleq \textit{ctx } L$$

Abella

Abella (Gacek 2008) is an interactive, tactics-based implementation of \mathcal{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 λ -calculus
- ▶ Tait-style weak normalizability proof

<http://abella.cs.umn.edu/>

Key parts of weak normalizability proof

The logical relation

$$\text{reduce } M \ i \triangleq (\Vdash \langle \text{of } M \ i \rangle) \wedge \text{halts } M$$

$$\begin{aligned} \text{reduce } M \ (\text{arr } A \ B) &\triangleq (\Vdash \langle \text{of } M \ (\text{arr } A \ B) \rangle) \wedge \text{halts } M \wedge \\ &\quad \forall N. (\text{reduce } N \ A \supset \text{reduce } (\text{app } M \ N) \ B) \end{aligned}$$

Substitution and freshness results

$$\text{subst nil } M \ M \triangleq \top$$

$$\begin{aligned} (\forall x. \text{subst } ((\text{of } x \ A) :: L) \ (R \ x) \ M) &\triangleq \\ \exists V. \text{reduce } V \ A \wedge (\Vdash \langle \text{value } V \rangle) &\wedge \text{subst } L \ (R \ V) \ M \end{aligned}$$

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^+) [Schürmann 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