Reasoning in Abella about Structural Operational Semantics Specifications

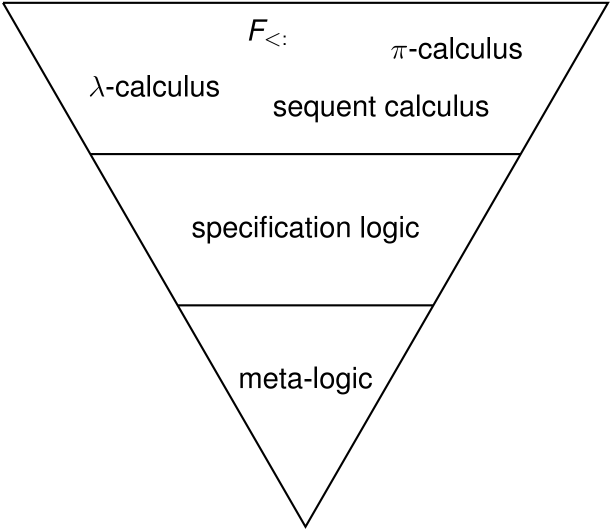
Andrew Gacek[1] Dale Miller[2] Gopalan Nadathur[1]

[1] Department of Computer Science and Engineering
University of Minnesota

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

LFMTP'08
June 23, 2008

Preview


$F_{<:}$
π-calculus
λ-calculus
sequent calculus
specification logic
meta-logic

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

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

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

This logic is a subset of the logic behind λProlog

```
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 λProlog 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

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

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

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

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

$$member\ A\ (A :: L) \triangleq \top$$
$$member\ A\ (B :: L) \triangleq member\ A\ L$$

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

$$
\begin{array}{ll}
seq_N\ L\ \langle A\rangle & \triangleq member\ A\ L \\
seq_{(s\ N)}\ L\ (B \wedge C) & \triangleq seq_N\ L\ B \wedge seq_N\ L\ C \\
seq_{(s\ N)}\ L\ (A \supset B) & \triangleq seq_N\ (A :: L)\ B \\
seq_{(s\ N)}\ L\ (\forall B) & \triangleq \nabla x. seq_N\ L\ (B\ x) \\
seq_{(s\ N)}\ L\ \langle A\rangle & \triangleq \exists b. prog\ A\ b \wedge seq_N\ L\ b
\end{array}
$$

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 $\exists n.nat\ n \wedge seq_n\ L\ G$
When L is *nil*, we write simply $\Vdash G$

Type substitution theorem:

$$\forall L, t_1, t_2, a, b. \nabla x. \quad (((of\ x\ a) :: L) \Vdash \langle of\ (t_1\ x)\ b \rangle) \wedge (L \Vdash \langle of\ t_2\ a \rangle) \supset (L \Vdash \langle of\ (t_1\ t_2)\ b \rangle)$$

Context permutation lemma:

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

Theorems about *seq*

Contexts admit weakening, contraction, and permutation

$$subset\ L_1\ L_2 \triangleq \forall X.member\ X\ L_1 \supset member\ X\ L_2$$

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

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)

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

Implicit properties of specifications

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

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

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

cntx L should enforce

- $L = (of\ x_1\ a_1) :: (of\ x_2\ a_2) :: \ldots :: (of\ x_n\ a_n) :: nil$
- Each x_i is atomic
- Each x_i 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

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

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

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

$$\mathit{cntx}\ \mathit{nil} \triangleq \top$$

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

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

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

Substitution and freshness results

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

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

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