4.6 Superposition in Higher-Order Logic

Problems originating from proof assistents

use higher-order logic,

but contain large first-order parts,

and in particular equality.

Can we extend the superposition calculus to higher-order logic?

Following Bentkamp et al. we proceed in three steps.

Step 1: Lambda-free HOL

We admit

- applied variables ($x\ b\ c$),
- unapplied or partially applied functions ($g\ f \approx h\ (f\ b\ c)$)

but exclude

- lambda abstractions
- first-class booleans (i. e., boolean expressions on the term level, rather than on the literal level)

This is also known as the *applicative fragment*.

In principle, one could encode it in FOL using constants and just one binary function symbol *app*.

FOL provers do not behave well on these formulas, though: Indexing data structures become almost useless. Term orderings do not behave in the expected way anymore.

First step:

Define a higher-order reduction ordering.

In addition to the usual properties of reduction orderings, one would like to have *compatibility with arguments*: $s \succ s'$ implies $s\ t \succ s'\ t$.

But this is difficult to achieve.

The calculus below works without this requirement.

A subterm t of s is called a *green subterm*, if $t = s$ or if $s = s'\,u_1 \ldots u_n$ and t is a green subterm of some u_i.

Notation: $s = s\langle t \rangle$.

Green subterms correspond to first-order subtermss.

If $t \succ t'$, then $s\langle t \rangle \succ s\langle t' \rangle$.

Second step:

Define the inference rules analogously to first-order superposition, but restrict to inferences that involve green subterms:

$$\frac{D' \lor t \approx t' \qquad C' \lor s\langle u \rangle \approx s'}{(D' \lor C' \lor s\langle t' \rangle \approx s')\sigma}$$

where $\sigma = \mathrm{mgu}(t, u)$.

Additionally:

New inference rule: ArgCong

$$\frac{C' \lor s \approx s'}{C' \lor s\;x \approx s'\;x}$$

Redundancy for inferences must be defined in such a way that the conclusion of ArgCong is not automatically redundant!

One more problem:

If $\succ$ is not compatible with arguments, we need occasionally superpositions at (but not below) variable positions. (The θ/θ' trick may not work anymore.)

Proof idea:

Use a two-fold lifting

- from HOL to ground HOL
- from ground HOL to ground FOL

Note: The Henkin interpretations contain only those functions that we can construct from the given ones.

In order to refute $b \not\approx x\;b$, our set of axioms should contain $id\;z \approx z$.

Step 2: Boolean-free HOL

We add lambda abstractions to the logic, but still exclude first-class booleans.

Need efficient HO unification procedure that enumerates a CSU (Vukmirović et al.)

Need dovetailing to interleave generation of further conclusions of inferences with clause selection.

Again: only inferences involving green subterms.

The definition of green subterms must be adapted, though:

A subterm t of s is called a *green subterm*, if $t = s$ or if $s = c\,u_1 \dots u_n$ for some constant c and t is a green subterm of some u_i.

(Subterms below applied variables are no longer green.)

Problem: Applying a grounding substitution θ that maps free variables to lambda expressions may fundamentally change the structure of a term t. Green subterms in $t\theta$ need not be instances of green subterms in t.

Examples:

Let $t = h\ (x\ b\ f)$ and $\theta = \{x \mapsto \lambda y\,z.\,g\ (z\ y)\}$. Then $t\theta = h\ (g\ (f\ b))$ contains the green subterm $f\ b$.

Let $t = \lambda x.\,f\ (y\ x)\ x$ and $\theta = \{y \mapsto \lambda z.\,c\}$. Then $t\theta = \lambda x.\,f\ c\ x = f\ c$ contains the green subterm $f\ c$.

Solution:

Need fluid inferences to ensure that all inferences between ground instances can be lifted:

$$\frac{D' \vee t \approx t' \qquad C' \vee s\langle u\rangle \approx s'}{(D' \vee C' \vee s\langle z\ t'\rangle \approx s')\sigma}$$

where $\sigma \in \mathrm{CSU}(z\ t, u)$.

Step 3: Full HOL

We add first-class booleans to the logic.

New problem: Performing a CNF transformation (including Skolemization) a priori is no longer sufficient.

Solution: Construct the HO superposition calculus on top of a *non-clausal* FO superposition calculus that performs CNF transformation steps on the fly.

We inherit the *hoisting* inferences of the non-clausal FO superposition calculus, e. g.

$$\frac{C\langle u\rangle}{(C\langle\bot\rangle \vee x \approx y)\sigma}$$

where $\sigma \in \mathrm{CSU}(u, x \approx y)$.

$$\frac{C\langle u\rangle}{(C\langle\top\rangle \vee x \approx y)\sigma}$$

where $\sigma \in \mathrm{CSU}(u, x \not\approx y)$.

Implementations:

Zipperposition (OCaml, full calculus)

E (C, parts of the calculus)

Alternative Approach

Extending an existing prover for FOL to handle lambda expressions requires substantial modifications of the architecture.

Alternative approach (Bhayat and Reger):

Every lambda expression can be encoded using a small set of *combinators*, e. g., $S = \lambda x\, y\, z.\, x\ z\ (y\ z)$, $K = \lambda x\, y.\, x$, $I = \lambda x.\, x$.

Use the lambda-free calculus together with the definitions of combinators.

Implemented in Vampire.

Literature

Peter B. Andrews: An Introduction to Mathematical Logic and Type Theory – To Truth Through Proof. 2nd edition, Springer, 2002.

Peter B. Andrews: Classical type theory. Handbook of Automated Reasoning, vol. II, pp. 965–1007. Elsevier and MIT Press, 2001.

Heiko Becker, Jasmin Christian Blanchette, Uwe Waldmann, Daniel Wand: A transfinite Knuth-Bendix order for lambda-free higher-order terms. CADE: 432–453, 2017.

Alexander Bentkamp, Jasmin Blanchette, Simon Cruanes, and Uwe Waldmann: Superposition for lambda-free higher-order logic. Logical Methods in Computer Science, 17(2):1:1–1:38, 2021.

Alexander Bentkamp, Jasmin Blanchette, Sophie Tourret, Petar Vukmirović, and Uwe Waldmann: Superposition with lambdas. Journal of Automated Reasoning, 65:893–940, 2021.

Alexander Bentkamp, Jasmin Blanchette, Sophie Tourret, and Petar Vukmirović: Superposition for higher-order logic. Journal of Automated Reasoning 67, Article number: 10, 2023.

Christoph Benzmüller, Dale Miller: Automation of higher-order logic. Computational Logic, Handbook of the History of Logic, vol. 9, pp. 215–254. Elsevier, 2014.

Jasmin Christian Blanchette, Uwe Waldmann, Daniel Wand: A lambda-free higher-order recursive path order. Foundations of Software Science and Computation Structures, FoSSaCS 2017, LNCS 10203, pp. 461–479, Springer, 2017.

Ahmed Bhayat, Giles Reger: A combinator-based superposition calculus for higher-order logic. Automated Reasoning, IJCAR 2020, Part I, LNCS 12166, pp. 278–296, Springer, 2020.

Gilles Dowek: Higher-order unification and matching. Handbook of Automated Reasoning, vol. II, pp. 1009–1062, Elsevier and MIT Press, 2001.

Melvin Fitting: Types, Tableaus, and Gödel's God. Studia Logica 81(3): 425–427, 2005.

Gérard P. Huet: A mechanization of type theory. IJCAI-73, pp. 139–146. William Kaufmann, 1973.

Gérard P. Huet: A unification algorithm for typed lambda-calculus. Theor. Comput. Sci. 1(1), 27–57, 1975.

D. C. Jensen, T. Pietrzykowski: Mechanizing ω-order type theory through unification. Theor. Comput. Sci. 3(2), 123–171, 1976.

Petar Vukmirović, Alexander Bentkamp, and Visa Nummelin: Efficient full higher-order unification. Logical Methods in Computer Science 17(4):18:1–18:31, 2021.

Petar Vukmirović, Jasmin Blanchette, Simon Cruanes, Stephan Schulz: Extending a brainiac prover to lambda-free higher-order logic. TACAS 2019, LNCS 11427, pp. 192–210. Springer, 2019.