Hierarchic Superposition With Weak Abstraction and the Beagle Theorem Prover

Peter Baumgartner
NICTA and ANU, Canberra

Uwe Waldmann
MPI für Informatik, Saarbrücken

Goal

Automated deduction in hierarchic combinations of specifications

Lists over integers

$(l \approx \mathbf{nil}) \vee (l \approx \mathbf{cons}(\mathbf{head}(l), \mathbf{tail}(l))$

$\neg(\mathbf{cons}(k, l) \approx \mathbf{nil})$

$\mathbf{head}(\mathbf{cons}(k, l)) \approx k$

$\mathbf{tail}(\mathbf{cons}(k, l)) \approx l$

The **inRange** predicate, e.g. **inRange([1,0,5], 6)**

$\mathbf{nRange}(l, n) \leftrightarrow (l \approx \mathbf{nil} \vee (0 \leq \mathbf{head}(l) < n \wedge \mathbf{inRange}(\mathbf{tail}(l), n)))$

Conjecture

$\forall$ l:list n:int $(\neg(\mathrm{l} \approx \mathbf{nil}) \rightarrow (\mathbf{inRange}(\mathrm{l}, \mathrm{n}) \rightarrow \mathbf{inRange}(\mathbf{cons}(\mathbf{head}(\mathrm{l}), \mathrm{l}), \mathrm{n})))$

$$\mathbf{LIA} + \mathbf{Lists/Arrays} + \mathbf{Hypotheses} \models \mathbf{Conjecture}\ ?$$

$$\mathbf{LIA} + \mathbf{Lists/Arrays} + \mathbf{Hypotheses} \not\models \mathbf{Conjecture}\ ?$$

Contents

Hierarchic Specifications

Background (BG) specification consists of

sorts, e.g. { **int** }

operators, e.g. { **0**, **1**, **-1**, **2**, **-2**, ..., -, +, >, **≥**}

models, e.g. linear integer arithmetic

Foreground (FG) specification extends BG specification by

new sorts, e.g. { list }

new operators, e.g.

{ **cons: int** × **list** ↦ **list**, **nil: list**,
length: list ↦ **int**, **a: list** }

first-order clauses, e.g.

{ **length(a)** ≥ **1**,
length(cons(*x*, *y*)) ≈ **length(*y*)+1** }

Hierarchic Specifications

Assumption

We have a decision procedure for quantified formulas over the BG specification

Goal

Check whether the hierarchic combination has models or not, using the BG decision procedure as a subroutine

Question

What is a model of the hierarchic combination?

Hierarchic Specifications

Models of hierarchic specifications

Must satisfy the FG clauses, and

must leave the interpretation of the BG sorts and operators unchanged (*conservative extension*):

- distinct BG elements may not be identified (*no confusion*), and
- no new elements may be added to BG sorts (*no junk*)

Fundamental problem 1

Absence of junk is not r.e.

⇒ Refutational completeness is only possible in certain cases

⇒ Require *sufficient completeness*

Sufficient Completeness

Sufficient Completeness

In every model of the FG clauses, every ground FG term that has a BG sort must be equal to some BG term

Example

$$(l \approx \mathbf{nil}) \vee (l \approx \mathbf{cons}(\mathbf{head}(l), \mathbf{tail}(l))$$
$$\neg(\mathbf{cons}(k, l) \approx \mathbf{nil})$$
$$\mathbf{head}(\mathbf{cons}(k, l)) \approx k$$
$$\mathbf{tail}(\mathbf{cons}(k, l)) \approx l$$

is not sufficiently complete:

take BG domain $\mathbb{Z} \cup \{ \text{NaN} \}$ and evaluate $\mathbf{head}(\mathbf{nil})$ to NaN

Adding $\mathbf{head}(\mathbf{nil}) \approx \mathbf{0}$ and $\mathbf{tail}(\mathbf{nil}) \approx \mathbf{nil}$ makes it sufficiently complete

Hierarchic Specifications

Fundamental Problem 2

We can pass only finite sets of formulas to the BG decision procedure

Second Requirement for Completeness

Compactness: If an infinite set of BG formulas is unsatisfiable, then it has a finite unsatisfiable subset

LIA with global symbolic constants $\boldsymbol{\alpha}$ (*parameters*) is not compact:

take $\{\ \boldsymbol{\alpha} \geq 0,\ \ \boldsymbol{\alpha} \geq 1,\ \ \boldsymbol{\alpha} \geq 2,\ \ \ldots\ \}$

LIA without parameters is compact

Calculi for Hierarchic Reasoning

If the FG clauses are ground

DPLL(T) + Nelson-Oppen

(Neither sufficient completeness nor compactness poses problems)

If the FG clauses are not ground

DPLL(T) + Nelson-Oppen + instantiation heuristics (CVC4, Z3,...)

Hierarchic superposition [Bachmair Ganzinger Waldmann 1994, Althaus Weidenbach Kruglov 2009, Weidenbach Kruglov 2012]

Model evolution with LIA constraints [B Tinelli 2008, 2011]

Sequent calculus [Rümmer 2008]

Theory instantiation [Korovin 2006]

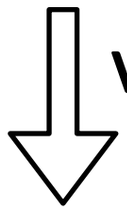
LASCA [Korovin Voronkov 2007]

Hierarchic superposition with weak abstraction [B Waldmann 2013]

Hierarchic Superposition with Weak Abstraction

Calculus Layout

Input clause set



Weak abstraction

Core calculus

- Superposition
- Close
- Weak abstraction
- Simplification
- Splitting
- Define

(Weak) Abstraction

Unification cannot detect "semantic equality" of BG terms

$$\frac{P(1+2) \qquad \neg P(2+1)}{?}$$

- Abstraction extracts BG terms in terms of new variables
- FG literals are subject to superposition inferences
- BG clauses are passed to the BG solver, in Close inferences

(Weak) Abstraction

Unification cannot detect "semantic equality" of BG terms

$$\frac{\mathsf{P}(1{+}2) \qquad \neg\mathsf{P}(2{+}1)}{?}$$

- Abstraction extracts BG terms in terms of new variables
- FG literals are subject to superposition inferences
- BG clauses are passed to the BG solver, in Close inferences

$$\mathsf{P}(X) \vee X \not\approx 1{+}2$$

(Weak) Abstraction

Unification cannot detect "semantic equality" of BG terms

$$\frac{\mathsf{P}(1+2) \qquad \neg\mathsf{P}(2+1)}{?}$$

- Abstraction extracts BG terms in terms of new variables
- FG literals are subject to superposition inferences
- BG clauses are passed to the BG solver, in Close inferences

$$\mathsf{P}(X) \vee X \not\approx 1+2 \qquad\qquad \neg\mathsf{P}(Y) \vee Y \not\approx 2+1$$

(Weak) Abstraction

Unification cannot detect "semantic equality" of BG terms

$$\frac{P(1+2) \qquad \neg P(2+1)}{?}$$

- Abstraction extracts BG terms in terms of new variables
- FG literals are subject to superposition inferences
- BG clauses are passed to the BG solver, in Close inferences

$$\frac{\underline{P(X)} \vee X \not\approx 1+2 \qquad \underline{\neg P(Y)} \vee Y \not\approx 2+1}{X \not\approx 1+2 \vee X \not\approx 2+1}\ \text{Sup}$$

(Weak) Abstraction

Unification cannot detect "semantic equality" of BG terms

$$\frac{P(1+2) \qquad \neg P(2+1)}{?}$$

- Abstraction extracts BG terms in terms of new variables
- FG literals are subject to superposition inferences
- BG clauses are passed to the BG solver, in Close inferences

$$\dfrac{\dfrac{\underline{P(X)} \vee X \not\approx 1+2 \qquad \underline{\neg P(Y)} \vee Y \not\approx 2+1}{X \not\approx 1+2 \vee X \not\approx 2+1}\ \text{Sup}}{\square}\ \text{Close}$$

Weak Abstraction

Weak Abstraction

Only non-variable BG terms that are

direct subterms of non-BG terms are abstracted out

Concrete numbers (0, 1, -1, 2, -2, ...) are never abstracted out

Example (α and β are BG constants)

$$g(1, \alpha, f(1)+(\alpha+1), Z) \approx \beta$$

$$\rightsquigarrow g(1, X, f(1)+\ Y,\quad Z) \approx \beta \vee X \not\approx \alpha \vee Y \not\approx (\alpha+1)$$

Properties (in relation to [BGW 94])

Extracts viewer terms: less detrimental to unification

Shorter clauses: preserves unit clauses more often (good for rewriting)

Always preserves sufficient completeness (see below)

Inference rules can destroy WA, hence need explicit WA of conclusion

Two Kinds of Variables

Abstraction Variables X, Y, Z

Stand for BG terms

→ Never unify with non-variable FG terms

Pro: BG terms are always smaller than FG terms

E.g., $\mathbf{f}(X) \approx \mathbf{g}(Y)$ is ordered from left to right if $\mathbf{f} > \mathbf{g}$

Con1: Subsumption does not work as expected

E.g., $\mathbf{P}(X)$ does not subsume $\mathbf{P}(\mathbf{f}(Y))$

Con2: Unexpectedly don't get refutations

E.g. $\mathbf{f(nil)} + 1 \not\approx Y + 1$

May even destroy sufficient completeness during abstraction

Ordinary variables fix these problems

Two Kinds of Variables

Ordinary Variables x, y, z

Stand for arbitrary BG-sorted terms

→ May also unify with non-variable FG terms

Con: viewer ordered equations

E.g., $\mathbf{f}(x) \approx \mathbf{g}(y)$ is not ordered from left to right even if $\mathbf{f} > \mathbf{g}$

Pro1: subsumption works as expected

E.g., $\mathbf{P}(x)$ subsumes $\mathbf{P}(\mathbf{f}(y))$

Pro2: mey get refutations even in absence of s.c.

E.g. $\mathbf{f}(\mathbf{nil}) + 1 \not\approx y + 1$

Always preserves sufficient completeness during abstraction (use ordinary variables only if abstracted term contains ordinary variables)

Two Kinds of Variables

What is the kind of variables in the input problem?

Ordinary variables: $\{ x \not\approx \mathbf{f(1)} \}$ has a refutation

Abstraction variables: $\{ X \not\approx \mathbf{f(1)} \}$ does not have a refutation

A: there is a trade-off, see above, so let the user decide.

In practice, most variables are abstraction variables, and ordinary variables are only used in additional *lemmas*:

Lemmas

Valid BG theory clauses, make BG knowledge available to FG reasoner

E.g. $\neg(x{<}x)$, $x{+}0 \approx x$, $\neg(x{<}y) \vee \neg(y{<}z) \vee x{<}z$

Used to simplify, e.g., $\mathbf{f(1)}{<}\mathbf{f(1)}$, $\mathbf{length(nil)}{+}0$

Definitions

In general, one cannot make an arbitrary hierarchic specification sufficiently complete by construction

We can, however, prevent that a *ground* BG-sorted FG term $\mathbf{t}$ is interpreted by a junk element:

- introduce a new parameter, i.e., a new BG constant $\boldsymbol{\alpha}_t$
- add the definition $\mathbf{t} \approx \boldsymbol{\alpha}_t$

This is a well-known preprocessing technique [KruglovWeidenbach 12]

However, in hierarchic superposition ground terms can show up in the middle of the saturation process

→ use introduction of definitions as an inference rule

$$\frac{\mathbf{f}(X) > 5 \vee X \not\approx 1+2}{\mathbf{f}(X) \approx \boldsymbol{\alpha}_{\mathbf{f}(1+2)} \vee X \not\approx 1+2}$$

Main Theoretical Results [B Waldmann CADE 2013]

Completeness 1

HSPWA is refutationally complete for compact BG specifications and sufficiently complete input clause sets

Completeness 2

HSPWA is refutationally complete for input clause sets where every BG-sorted term is ground

The Beagle Prover

- Full implementation of the calculus above
  - Lemmas, ordinary/abstraction variables, definitions, splitting
  - Discount/otter loop, demodulation, subsumption
  - LPO/KBO, W/A ratio
  - Cautious and aggressive simplification
    e.g. $1+(2+a) \not\approx 1+x$ simplifies to $3+a \not\approx 1+x$
- Front-end for TPTP TFA and SMT-Lib languages
- Background reasoners
  - LRA: Fourier/Motzkin, Simplex
  - LIA: Cooper QE, Branch and bound
- Written in Scala, easy to install
  http://users.cecs.anu.edu.au/~baumgart/systems/beagle/
- Team: PB, A Bauer, J Bax, T Cosgrove
- Companion system: SMTtoTPTP
  http://users.cecs.anu.edu.au/~baumgart/systems/smttotptp/

User Experience

There is a number in [a,...,a+2] that is divisible by 3

```
$ beagle ARI595=1.p

This is beagle, version 0.7.1 (2/10/2013)

Input formulas
==============
¬((∀ Zª:$int (((a ≤ Zª) ∧ (Zª ≤ (a + 2))) ⇒ p(Zª))) ⇒ (∃ Xª:$int p((3·Xª))))

Clause set signature
====================
Background sorts: {□, ℤ, ℝ}
Foreground sorts: {$i, $o, $tType}
Background operators:
    $greatereq: ℤ × ℤ → $o
    :
    a: ℤ
Foreground operators:
    $true: $o
    p: ℤ → $o
    $false: $o
```

User Experience

```
Precedence among foreground operators: p > $true > $false

Clause set
==========
p(Zᵃ) ∨ ¬(Zᵃ ≤ (a + 2)) ∨ ¬(a ≤ Zᵃ)
¬p((3·Xᵃ))

Background sorts used in clause set: ℤ
Used background theory solver: cooper-clauses

Proving...
p(Zᵃ) ∨ ¬(Zᵃ ≤ (2 + a)) ∨ ¬(a ≤ Zᵃ)
¬p((3·Xᵃ))
¬((3·X_13ᵃ) ≤ (2 + a)) ∨ ¬(a ≤ (3·X_13ᵃ))
¬(3|a)
¬(3|(2 + a))
¬(3|(1 + a))

SZS status Theorem for ARI595=1.p

Inference rules
----------------------
...
```

Beagle on TPTP

LIA-Theorems in TPTP	337
Full Abstraction [BGW94]	242 proved
Weak Abstraction	251 proved
WA + Definitions	297 proved
WA + Definitions + Aggressive Simplification	303 proved

Two more theorems can only be proved when BG axioms are added, but adding BG axioms is (obviously) a double-edged sword and not generally helpful

Proving Infinite Satisfiability [B Bax LPAR-19]

- Given the LIST axioms over integers
- Suppose a set HYP defining functions/relations on lists
  E.g. **length**, **in**, **inRange**, **count**, **append**
- Suppose we know that LIST ∪ HYP is satisfiable (by construction)
- Then, to disprove a conjecture CON, i.e.

  $$\text{LIST} \cup \text{HYP} \not\models \text{CON}$$

  it suffices to prove

  $$\text{LIST} \cup \text{HYP} \models \neg\text{CON}$$

- Same for ARRAY
- Use this method in the following result tables, for all provers
  - Directly establishing countersatisfiability does not work at all

Experiments with LIST

Problem	Beagle	Spass+T	Z3
$\mathsf{inRange}(4, \mathsf{cons}(1, \mathsf{cons}(5, \mathsf{cons}(2, \mathsf{nil}))))$	6.2	0.3	0.2
$n > 4 \Rightarrow \mathsf{inRange}(n, \mathsf{cons}(1, \mathsf{cons}(5, \mathsf{cons}(2, \mathsf{nil}))))$	7.2	0.3	0.2
$\mathsf{inRange}(n, \mathsf{tail}(l)) \Rightarrow \mathsf{inRange}(n, l)$	3.9	0.3	0.2
$\exists\, n_{\mathbb{Z}}\ l_{\mathsf{LIST}}\,.\, l \not\approx \mathsf{nil} \wedge \mathsf{inRange}(n, l) \wedge n - \mathsf{head}(l) < 1$	2.7	0.3	0.2
$\mathsf{inRange}(n, l) \Rightarrow \mathsf{inRange}(n-1, l)$	8.2	0.3	>60
$l \not\approx \mathsf{nil} \wedge \mathsf{inRange}(n, l) \Rightarrow n - \mathsf{head}(l) > 2$	2.8	0.3	0.2
$n > 0 \wedge \mathsf{inRange}(n, l) \wedge l' = \mathsf{cons}(n-2, l) \Rightarrow \mathsf{inRange}(n, l')$	4.5	5.2	0.2

Problems 5 and 7 require "ordinary variables" and "cautious simplification" (the most complete parameter setting)

Experiments with LIST

Problem	Beagle	Spass+T	Z3
$\mathsf{length}(l_1) \approx \mathsf{length}(l_2) \Rightarrow l_1 \approx l_2$	4.3	9.0	0.2
$n \geq 3 \wedge \mathsf{length}(l) \geq 4 \Rightarrow \mathsf{inRange}(n, l)$	5.4	1.1	0.2
$\mathsf{count}(n, l) \approx \mathsf{count}(n, \mathsf{cons}(1, l))$	2.5	0.3	>60
$\mathsf{count}(n, l) \geq \mathsf{length}(l)$	2.7	0.3	>60
$l_1 \not\approx l_2 \Rightarrow \mathsf{count}(n, l_1) \not\approx \mathsf{count}(n, l_2)$	2.4	0.8	>60
$\mathsf{length}(\mathsf{append}(l_1, l_2)) \approx \mathsf{length}(l_1)$	2.1	0.3	0.2
$\mathsf{length}(l_1) > 1 \wedge \mathsf{length}(l_2) > 1 \Rightarrow \mathsf{length}(\mathsf{append}(k, l)) > 4$	37	>60	>60
$\mathsf{in}(n_1, l_1) \wedge \neg\mathsf{in}(n_2, l_2) \wedge l_3 \approx \mathsf{append}(l_1, \mathsf{cons}(n_2, l_2)) \Rightarrow \mathsf{count}(n, l_3) \approx \mathsf{count}(n, l_1)$	>60 (6.2)	9.1	>60

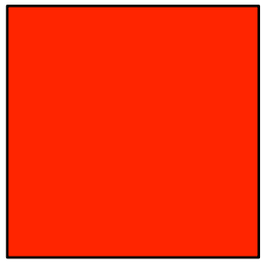
The last problem is provable only with "abstraction variables"

Experiments with ARRAY

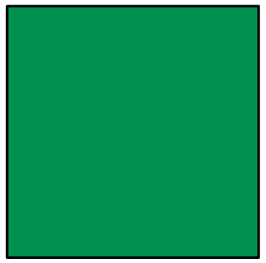
Problem	Beagle	Spass+T	Z3
$n \geq 0 \Rightarrow \mathsf{inRange}(a, \mathsf{max}(a, n), n)$	1.40	0.16	u
$\mathsf{distinct}(\mathsf{init}(n), i)$	0.98	0.15	u
$\mathsf{read}(\mathsf{rev}(a, n+1), 0) = \mathsf{read}(a, n))$	>60	>60(0.27)	>60
$\mathsf{distinct}(a, n) \Rightarrow \mathsf{distinct}(\mathsf{rev}(a, n))$	>60	0.11	0.36
$\exists\, n_{\mathbb{Z}} \,.\, \neg\mathsf{sorted}(\mathsf{rev}(\mathsf{init}(n), m), m)$	>60	0.16	u
$\mathsf{sorted}(a, n) \wedge n > 0 \Rightarrow \mathsf{distinct}(a, n)$	2.40	0.17	0.01

The first and the last problem is provable only with "ordinary variables"

Colors



Foreground red



Foreground green

Code Regular text