Superposition: Extensions

Extensions and improvements:

simplification techniques,

selection functions (when, what),

redundancy for inferences,

constraint reasoning,

decidable first-order fragments.

Theory Reasoning

Superposition vs. resolution + equality axioms:

specialized inference rules,
thus no inferences with theory axioms,

computation modulo symmetry,

stronger ordering restrictions,

no variable overlaps,

stronger redundancy criterion.

Theory Reasoning

Similar techniques can be used for other theories:

transitive relations,

dense total orderings without endpoints,

commutativity,

associativity and commutativity,

abelian monoids,

abelian groups,

divisible torsion-free abelian groups.

Part 7: Outlook

Further topics in automated reasoning.

7.1 Satisfiability Modulo Theories (SMT)

CDCL checks satisfiability of propositional formulas.

CDCL can also be used for ground first-order formulas without equality:

Ground first-order atoms are treated like propositional variables.

Truth values of $P(a)$, $Q(a)$, $Q(f(a))$ are independent.

Satisfiability Modulo Theories (SMT)

For ground formulas with equality, independence is lost:

If $b \approx c$ is true, then $f(b) \approx f(c)$ must also be true.

Similarly for other theories, e. g. linear arithmetic: $b > 5$ implies $b > 3$.

We can still use CDCL, but we must combine it with a decision procedure for the theory part T:

$M \models_T C$: M and the theory axioms T entail C.

Satisfiability Modulo Theories (SMT)

New CDCL rules:

T-Propagate:

$$M \parallel N \;\Rightarrow_{\text{CDCL(T)}}\; M\,L \parallel N$$

if $M \models_T L$ where L is undefined in M and L or $\overline{L}$ occurs in N.

T-Learn:

$$M \parallel N \;\Rightarrow_{\text{CDCL(T)}}\; M \parallel N \cup \{C\}$$

if $N \models_T C$ and each atom of C occurs in N or M.

Satisfiability Modulo Theories (SMT)

T-Backjump:

$$M\ L^{\mathrm{d}}\ M' \parallel N \cup \{C\} \ \Rightarrow_{\mathrm{CDCL(T)}}\ M\ L' \parallel N \cup \{C\}$$

if $M\ L^{\mathrm{d}}\ M' \models \neg C$
and there is some "backjump clause" $C' \vee L'$ such that
$N \cup \{C\} \models_T C' \vee L'$ and $M \models \neg C'$,
L' is undefined under M, and
L' or $\overline{L'}$ occurs in N or in $M\ L^{\mathrm{d}}\ M'$.

7.2 Sorted Logics

So far, we have considered only unsorted first-order logic.

In practice, one often considers many-sorted logics:

$\mathit{read}/2$ becomes $\mathit{read} : \mathit{array} \times \mathit{nat} \to \mathit{data}$.

$\mathit{write}/3$ becomes $\mathit{write} : \mathit{array} \times \mathit{nat} \times \mathit{data} \to \mathit{array}$.

Variables: $x : \mathit{data}$

Only one declaration per function/predicate/variable symbol.

All terms, atoms, substitutions must be well-sorted.

Sorted Logics

Algebras:

Instead of universe $U_{\mathcal{A}}$, one set per sort: $\mathit{array}_{\mathcal{A}}$, $\mathit{nat}_{\mathcal{A}}$.

Interpretations of function and predicate symbols correspond to their declarations:

$\mathit{read}_{\mathcal{A}} : \mathit{array}_{\mathcal{A}} \times \mathit{nat}_{\mathcal{A}} \rightarrow \mathit{data}_{\mathcal{A}}$

Sorted Logics

Proof theory, calculi, etc.:

Essentially as in the unsorted case.

More difficult:

Subsorts

Overloading

Better treated via relativization:

$\forall x_S \; \phi \Rightarrow \forall y \; S(y) \rightarrow \phi\{x_S \mapsto y\}$

7.3 Splitting

Tableau-like rule within resolution to eliminate variable-disjoint (positive) disjunctions:

$$\frac{N \cup \{C_1 \vee C_2\}}{N \cup \{C_1\} \quad | \quad N \cup \{C_2\}}$$

if $var(C_1) \cap var(C_2) = \emptyset$.

Split clauses are smaller and more likely to be usable for simplification.

Splitting tree is explored using intelligent backtracking.

7.4 Integrating Theories into Superposition

Certain kinds of theories/axioms are

important in practice,

but difficult for theorem provers.

So far important case: equality

but also: transitivity, arithmetic...

Integrating Theories into Superposition

Idea: Combine Superposition and Constraint Reasoning.

Superposition Left Modulo Theories:

$$\frac{\Lambda_1 \parallel C_1 \vee t \approx t' \qquad \Lambda_2 \parallel C_2 \vee s[u] \not\approx s'}{(\Lambda_1, \Lambda_2 \parallel C_1 \vee C_2 \vee s[t'] \not\approx s')\sigma}$$

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

...

Advertisements

Interested in Bachelor/Master/PhD Thesis?

Automated Reasoning

contact Christoph Weidenbach

(MPI-INF, MPI-SWS Building, 6th floor)

Hybrid System Verification

contact Uwe Waldmann

Arithmetic Reasoning (Quantifier Elimination)

contact Thomas Sturm

Advertisements

Next semester:

Automated Reasoning II

Content: Integration of Theories (Arithmetic)

Lecture: Block Course

Tutorials: TBA