Combination of Decision Procedures*

Franz Baader

University of Technology Aachen

Germany

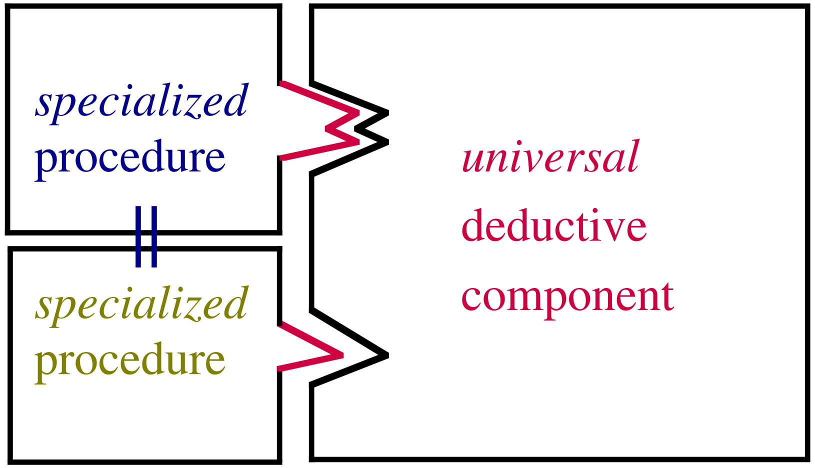
- Motivation: constraint solving in Automated Theorem Proving and Logic Programming
- Combination of unification algorithms
- Logical and algebraic analysis of the results
- Extension to more general constraint solvers

* This is joint work with Klaus U. Schulz, University of Munich

equational unification

ordering constraints

data structures (lists, sets)



resolution-based theorem provers

Knuth-Bendix completion

Logic Programming languages

Combination problems

- Coupling of specialized procedures with the universal component
- Coupling of different specialized procedures

Logic Programming language with arithmetic component

Environment-friendly freight forwarding company:
Whenever possible, use rail instead of trucks.

```
directly-connected(Aachen,Cologne)
        .
        .
        .
connected(X,Y) :- directly-connected(X,Y)

connected(X,Y) :- directly-connected(X,Z), connected(Z,Y)


transport-with(X,Y,rail) :- connected(X,Y)
```

Additional economic constraints

Costs should not be significantly higher and transportation time should not be significantly longer compared to transportation by trucks

Calculations involving (e.g. rational, real) numbers

logical encoding of the theory of real numbers is too inefficient → use efficient specialized procedures; e.g., procedures for solving linear equations and inequations (Simplex)

Other specialized procedures relevant in this context:

Integration of data structures such as sets and lists (e.g. Prolog III)

Resolution method

Goal refutation proof by inferring the empty clause

Resolution rule

$$\frac{P(s) \vee C_1 \qquad \neg P(t) \vee C_2}{\sigma(C_1 \vee C_2)}$$

σ is a unifier of s and t:

$\sigma(s) = \sigma(t)$

It is sufficient to compute the most general unifier: every unifier is an instance of the mgu

Building-in specialized procedures

by modifying the unification component, e.g.,

- unification modulo equational theories
- replace unification by constraint solving (e.g. CLP(R))

Unification modulo equational theories

Plotkin 1972: resolution-based theorem provers waste a lot of time by applying axioms like associativity and commutativity.

Example

f associative

Apply the resolution rule to

$$P(f(x,x)) \text{ and } \neg P(f(x_1,\ldots,f(x_n,f(x_1,\ldots,f(x_{n-1},x_n)\ldots))\ldots))$$

Automated theorem prover	Mathematician
Many ways of re-arranging parentheses in the right term "most of them" don't unify with f(x,x)	work modulo associativity, i.e. consider words instead of terms: $\neg P(x_1\ldots x_n x_1\ldots x_n)$ $P(xx)$

E-unification

Set of equational axioms	$E = \{ \ldots, s = t, \ldots \}$
Induced equational theory	$s =_E t$ iff $E \models s = t$
E-unification problem	$\Gamma = \{s_1 = t_1, \ldots, s_n = t_n\}$
E-unifier of Γ	substitution σ such that $\sigma(s_1) =_E \sigma(t_1), \ldots, \sigma(s_n) =_E \sigma(t_n)$
Complete set of E-unifiers $cU_E(\Gamma)$	Every E-unifier θ is an instance of an element σ of $cU_E(\Gamma)$, i.e., there is λ with $\theta(x) =_E \lambda(\sigma(x))$ for all variables x in Γ.

plays role of
most general unifier

Rheinisch-
Westfälische
Technische
Hochschule
Aachen
RWTH

Associativity

$A = \{f(x,f(y,z)) = f(f(x,y),z)\}$

Unification problem $\Gamma = \{f(x,a) = f(a,x)\}$

$\sigma_1 = \{x \to a\}$	is syntactic unifier and A-unifier
$\sigma_2 = \{x \to f(a,a)\}$	is A-unifier, but not a syntactic unifier

Plotkin

procedure that enumerates complete sets of A-unifiers; these sets may be infinite.

In the example: $cU_A(\Gamma) = \{\sigma_n \mid n \geq 1\}$

Makanin

A-unifiability is decidable.

Rheinisch-
Westfälische
Technische
Hochschule
Aachen
RWTH

Constraint solving instead of unification

Instead of computing a complete set of unifiers, just test solvability of the unification problem.

Example

f associative

Axioms: $Q(x) \vee P(f(x,a),f(a,x))$, $\neg P(y,y)$

Assumption: $\neg Q(b)$

$$\frac{Q(x) \vee P(f(x,a),f(a,x)) \qquad \neg P(y,y)}{Q(x) \mid f(x,a) = y = f(a,x)}$$

resolution with $\neg Q(b)$ not possible, since $\{f(x,a) = y = f(a,x),\ x = b\}$ is not solvable

Rheinisch-Westfälische Technische Hochschule Aachen
RWTH

Requirements on the unification algorithm

when building-in E-unification
into a resolution-based theorem prover

One needs an algorithm for general E-unification, i.e., the terms to be unified may contain additional free function symbols.

Example

A = {f(x,f(y,z)) = f(f(x,y),z)} contains only f.
Free function symbols are generated by Skolemization.

$\exists x\ \forall y\colon f(x,y) = y \longrightarrow \forall y\colon f(e,y) = y$

Skolemization

$\forall y\ \exists z\colon f(z,y) = e \longrightarrow \forall y\colon f(i(y),y) = e$

Makanin's decision procedure cannot deal with free function symbols.

Rheinisch-
Westfälische
Technische
Hochschule
Aachen
RWTH

Combination of unification algorithms

How can unification algorithms for the theories E and F be used to construct an algorithm for the combined theory $E \cup F$?

Examples

- Building-in an associative symbol f and a commutative symbol g.
- Going from A-unification with free constants to general A-unification corresponds to the combination of an A-unification algorithm with an algorithm for syntactic unification.

Disjointness

of the signatures of E and F

- For arbitrary non-disjoint signatures there cannot exists general combination methods.

Combination results

computation of complete sets of unifiers
equational theories over disjoint signatures

- First solved for the combination of several associative-commutative symbols and free symbols [Stickel75, Fages84, Herold&Siekmann87].
- Generalized to classes of theories whose axioms satisfy certain syntactic restrictions [Kirchner85, Tiden86, Herold86, Yelick87, ...].

Schmidt-Schauß89

solves the problem in a rather general way

+ No syntactic restrictions on the axioms of the equational theory.
+ Requirements are of an algorithmic nature: in addition to an algorithm for unification with constants one needs a "constant elimination procedure."
− Logical/algebraic meaning of this requirement is not clear.
− The method cannot be used to combine decision procedures.

Combination of decision procedures

[Baader&Schulz92]

Unification modulo $E \cup F$ is decidable, provided that

- E and F are equational theories over disjoint signatures,
- unification with linear constant restrictions is decidable for E and F.

Linear constant restrictions

linear ordering $<$ on the variables and constants of the problem

$x < c$: the constant c must not occur in the image of the variable x

$\{f(x) = f(b)\}$: does not have a syntactic unifier under the restriction $x < b$

Consequences

- General A-unification is decidable: Makanin's algorithm can "easily" be extended to an algorithm for A-unification with lcr [Schulz91].
- Modularity result: the combination method yields a decision procedure for unification with lcr in the combined theory.
- Complexity result: NP-decidability can be lifted to the combined theory.
- Complete sets: the combination method can also be used to combine algorithms computing complete sets of unifiers. The combination result of Schmidt-Schauß can be obtained as a corollary.

Rheinisch-
Westfälische
Technische
Hochschule
Aachen
RWTH

decomposition algorithm

Γ_0

unification problem for $E \cup F$
$sig(E) \cap sig(F) = \emptyset$

transformation rules

- standard transformation steps used in many combination procedures (variable abstraction, variable identification, ...)
- New: choose a linear ordering
- some of the steps are non-deterministic (responsible for complexity of algorithm)

Γ_1

unification problem with lcr for E

Γ_2

unification problem with lcr for F

Rheinisch-Westfälische Technische Hochschule Aachen
RWTH

f associative

$g(f(y,y)) = g(x),\ g(x) = g(y)$

g free

variable abstraction:
generates "pure" terms

$z = f(y,y)$	$g(z) = g(x),\ g(x) = g(y)$

choose theory for variables:
avoids incompatible instantiations

$z = f(y,y)$	$g(z) = g(x),\ g(x) = g(y)$

choose variable identification:
avoids incompatible free constants

$x = f(y,y)$	$g(x) = g(x),\ g(x) = g(y)$

linear constant restriction:
avoids cyclic dependencies

Analysis of results

- more abstract
- less technical formulation

goal →

- better understanding
- simpler proofs
- easier to generalize

Two approaches

- logical and algebraic characterizations of unification with linear constant restrictions.
- combination of algebras and solution structures instead of union of theories.

A logical view on unification

$\Gamma = \{s_1 = t_1, \ldots, s_n = t_n\}$ is E-unifiable	$\longleftrightarrow$	$E \models \exists \underline{x}: s_1 = t_1 \wedge \ldots \wedge s_n = t_n$
elementary E-unification is decidable	$\longleftrightarrow$	validity in E of existential positive formulae is decidable

elementaty unification:
no free symbols

Rheinisch-
Westfälische
Technische
Hochschule
Aachen
RWTH

Logical characterization of linear constant restrictions

equivalence result

unification with lcr decidable	validity in the positive theory decidable	general unification decidable

example for E = Ø

$\{f(x) = f(b)\}$ with restriction $x < b$	$\exists x\ \forall y_b\ f(x) = f(y_b)$	$\{f(x) = f(g(x))\}$
not unifiable	*not valid*	*not unifiable*

Rheinisch-
Westfälische
Technische
Hochschule
Aachen
RWTH

Algebraic reformulation of the combination results

- The equational theory E defines the variety $V(E) = \{\mathcal{A} \mid E \models \mathcal{A}\}$.
- The quotient term algebra $T(\Sigma,X)/=_E$ is free in V(E) with generators X.
- E-unification is solving equations in the E-free algebra $T(\Sigma,V)/=_E$ with countably infinite set of generators V.
- The free algebra $T(\Sigma,V)/=_E$ is canonical for the positive theory of E, i.e., a positive formula is valid in E iff it is valid in $T(\Sigma,V)/=_E$.

positive theories of $T(\Sigma,V)/=_E$ and $T(\Delta,V)/=_F$ decidable	$\longrightarrow$	positive theory of $T(\Sigma \cup \Delta,V)/=_{(E \cup F)}$ decidable

E and F equational theories over disjoint signatures Σ and Δ

Generalization of results

More general constraints

than purely equational constraints

- Allow for additional predicate symbols (different from =) in the signature and consider free structures instead of free algebras.
- Modularity result for decidability of positive theories of free structures [Baader&SchulzRTA95].

More general solution structures

than free algebras/structures

- Many of the solution structures considered in constraint programming are not free:
  - rational trees in CLP-languages (e.g. Prolog III)
  - feature structures in computational linguistics (e.g. Life, Oz)

Combination of structures

We are looking for (abstract algebraic) characterizations of interesting classes of structures for which there is a general combination construction $\circledast$ such that the following holds:

- If $\mathcal{A}$ is a Σ-structure and $\mathcal{B}$ a Δ-structure (for disjoint signatures Σ and Δ), then $\mathcal{A} \circledast \mathcal{B}$ is a $(\Sigma \cup \Delta)$-structure.
- The positive theory of $\mathcal{A} \circledast \mathcal{B}$ is decidable, provided that the positive theories of $\mathcal{A}$ and of $\mathcal{B}$ are decidable.
- This decidability result can be shown using a variant of the decomposition algorithm.

When is $\mathcal{A} \circledast \mathcal{B}$ a "reasonable" combination of $\mathcal{A}$ *and* $\mathcal{B}$?

Rheinisch-
Westfälische
Technische
Hochschule
Aachen
RWTH

The free amalgamated product

arbitrary structures
no disjointness restriction

is characterized by a universal property:

common part

$$
\begin{tikzcd}
 & \mathcal{A}^{\Sigma} \arrow[dr] \arrow[drr] & & \\
\mathcal{G}^{\Sigma\cap\Delta} \arrow[ur] \arrow[dr] & & \mathcal{A} \circledast \mathcal{B} \arrow[r, "!"] & \mathcal{D}^{\Sigma\cup\Delta} \\
 & \mathcal{B}^{\Delta} \arrow[ur] \arrow[urr] & &
\end{tikzcd}
$$

$\mathcal{A} \circledast \mathcal{B}$ and $\mathcal{D}^{\Sigma\cup\Delta}$ belong to a class of "admissible" structures $\mathrm{Adm}(\mathcal{A},\mathcal{B})$.

Need not exist for arbitrary structures!

Results

- If the free amalgamated product exists, then it is unique up to isomorphism.
- For free algebras, the free amalgamated product always exists:

$$\boxed{T(\Sigma,X)/{=_E} \circledast T(\Delta,X)/{=_F} \cong T(\Sigma \cup \Delta,X)/{=_{(E \cup F)}}}$$

where $T(\Sigma\cap\Delta,X)$ is the common part and $V(E\cup F)$ is the admissible class.

Questions

- Is there a larger class of structures for which the free amalgamated product always exists?
- Can this product be obtained by an explicit construction?
- Is our combination method for decision procedures applicable?

Internal characterization of free structures

The structure $\mathcal{A}$ is free over the countably infinite set of generators V (for some variety V(E)) iff

1) $\mathcal{A}$ is generated by V.

2) Every mapping f: V $\rightarrow$ A can be extended to an endomorphism of $\mathcal{A}$.

Generalization

- Keep 2): important in the proof of correctness of the combination method.
- Weaken 1): "generated by" is replaced by "stabilized by".

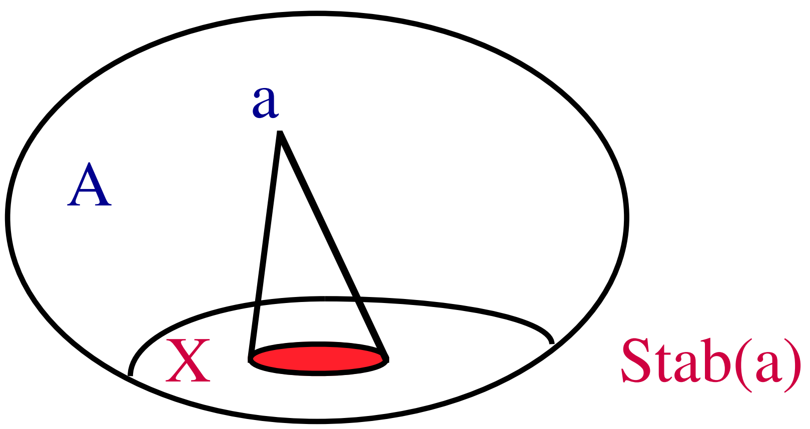
Quasi-free structures

[Baader&SchulzCP95,TCS98]

The countably infinite structure $\mathcal{A}$ is called quasi-free over the countably infinite set of "atoms" X iff

1) For every $a \in A$ there exists a finite set $Stab(a) \subseteq X$ such that endomorphisms of $\mathcal{A}$ that agree on Stab(a) also agree on a.

2) Every mapping $f: X \rightarrow A$ can be extended to an endomorphism of $\mathcal{A}$.

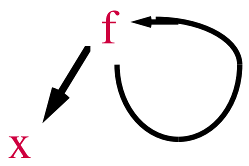


Rheinisch-Westfälische Technische Hochschule Aachen
RWTH

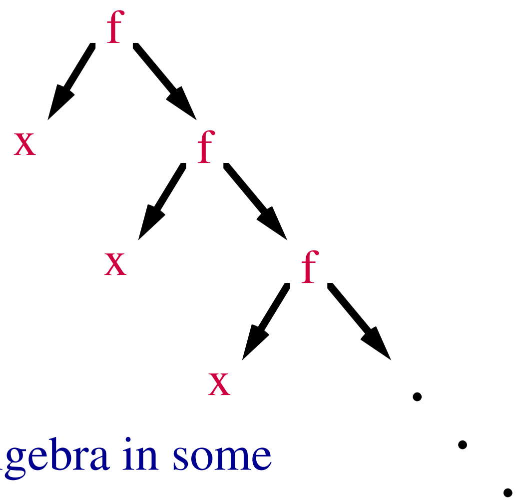
The algebra of rational trees

is quasi-free, but not free

- Elements are countable (finite or infinite) trees containing only finitely many different subtrees.



solution of
$y = f(x,y)$



- Used as solution structure in place of the term algebra in some Logic Programming languages (e.g. Prolog III).
- Ad-hoc approaches for combination with data structures such as sets and lists.

Results

for quasi-free structures

- Investigation of the algebraic and logical properties of quasi-free structures.
- Definition of an explicit amalgamation construction for quasi-free structures over disjoint signatures.
- This construction yields the free amalgamated product.
- It allows for a purely algebraic proof of correctness of the decomposition algorithm.
- More abstract (less technical) understanding of why our combination method works.

Theorem

[Baader&SchulzTCS98]

Let $\mathcal{A}$ and $\mathcal{B}$ be quasi-free structures over disjoint signatures.

1) The free amalgamated product $\mathcal{A} \circledast \mathcal{B}$ of $\mathcal{A}$ and $\mathcal{B}$ always exists.

2) If the positive theories of $\mathcal{A}$ and $\mathcal{B}$ are decidable, then the positive theory of $\mathcal{A} \circledast \mathcal{B}$ is decidable as well.

This combination result applies to important solution structures such as the algebra of rational trees, feature structures, and hereditarily finite well-founded or non-well-founded sets and lists.

Conclusion

- Combination of decision procedures for unification modulo equational theories.
- General approach for combining solution structures: free amalgamated product.
- Definition of the class of quasi-free structures:
  - Generalization of free structures.
  - Allow for an explicit amalgamation construction.
  - Combination of decision procedures for the positive theories of quasi-free structures.

Rheinisch-
Westfälische
Technische
Hochschule
Aachen
RWTH