Building and Combining Unification and Matching Procedures: A Hierarchical Approach[1]

Christophe Ringeissen

{Inria, Université de Lorraine, CNRS, LORIA}, Nancy, France

Summer Workshop in Mathematics, UnB, February 2024

[1] Joint work with Serdar Erbatur (UT Dallas) and Andrew Marshall (Univ. Mary Washington)

Unification Problems

Unification problem: finite set of equations between terms

$$\{x + b = a + y \ , \ y = b\}$$

Solution to a unification problem: a substitution of variables making each equation true

$$\{x \mapsto a, y \mapsto b\}$$

Applications: in logic programming, theorem proving, deductive verification to perform a deduction/computation

➽ Unification is used when applying resolution between clauses

Syntactic Unification and Equational Unification

Syntactic unification: $s = t$ is true iff s and t are identical

Equational unification: $s = t$ is true iff s and t are equal modulo an equational theory, e.g.,

Associativity-Commutativity:
$AC(+) = \{X + Y = Y + X,\ X + (Y + Z) = (X + Y) + Z\}$

Abelian Groups:
$AG(+) = AC(+) \cup \{X + 0 = X,\ X + (-X) = 0\}$

Exclusive Or:
$XOR(\oplus) = AC(\oplus) \cup \{X \oplus 0 = X,\ X \oplus X = 0\}$

Equational unification is undecidable in general, but decidable for some particular equational theories such as the ones above

Equational Matching

Equational unification: solving equations $s =^?_{\mathcal{E}} t$ modulo an equational theory $\mathcal{E}$ where s and t are arbitrary terms

Equational matching: solving equations $s =^?_{\mathcal{E}} t$ modulo an equational theory $\mathcal{E}$ where s or t is **ground**

Applications: (equational) rewriting, rule-based programming, simplification in theorem proving,

Equational matching/unification is undecidable in general, but decidable for particular equational theories $\mathcal{E}$ possibly including

- Associativity: $A(*) = \{X * (Y * Z) = (X * Y) * Z\}$
- Commutativity: $C(*) = \{X * Y = Y * X\}$
- Associativity-Commutativity: $AC(*) = A(*) \cup C(*)$

Example: $x * y = b * b * d \quad \vdash_{AC(*)-\text{Match}} \quad x = b * b, y = d$

$\vdash_{AC(*)-\text{Match}} \quad x = b, y = b * d$

$\ldots$

Rule-based Unification

Goal: Design a unification procedure as inference system transforming equational problems

$$\Gamma = \{s_1 = t_1, \ldots, s_n = t_n\}$$

until reaching solved forms, and satisfying the following properties:

sound If $\Gamma \vdash \Gamma'$, then any unifier of Γ' is a unifier of Γ

complete If $\Gamma \vdash \Gamma'$, then any unifier of Γ is a unifier of Γ'

terminating if $\Gamma \vdash \Gamma'$, then $c(\Gamma) > c(\Gamma')$, where c is a measure associated to equational problems, and $>$ is an ordering with no infinite decreasing chain

Rule-based Unification: Solved Forms

An equational problem is irreducible with respect to a rule-based unification procedure if and only if it is a solved form.

Two kinds of solved forms:

Tree solved form $\Gamma = \{x_1 = t_1, \ldots, x_n = t_n\}$
where for $i = 1, \ldots, n$, x_i is a variable occurring once in Γ

Dag solved form $\{x_1 = t_1, \ldots, x_n = t_n\}$
where for $i, j = 1, \ldots, n$,
$i \neq j$ implies x_i and x_j are distinct variables,
and $i \leq j$ implies x_i does not occur in t_j

NB: a solved form yields a most general unifier.

Syntactic vs. Equational Unification

The following decomposition rule is sound and complete for syntactic unification:

$$\textbf{Dec} \quad \{f(s_1,\ldots,s_n) = f(t_1,\ldots,t_n)\} \cup \Gamma \\ \vdash \{s_1 = t_1,\ldots,s_n = t_n\} \cup \Gamma$$

Dec remains sound for equational unification, but additional transformation rules are needed to retrieve completeness.

For example, when f is a commutative binary symbol:

$$\textbf{Mut} \quad \{f(s_1,s_2) = f(t_1,t_2)\} \cup \Gamma \\ \vdash \{s_1 = t_2,\ldots,s_2 = t_1\} \cup \Gamma$$

$\{\textbf{Dec},\textbf{Mut}\}$ leads to a sound, complete and terminating commutative unification procedure.

Question: can we generalize this idea of *mutation* rule to other equational theories?

Syntactic Theories

Definition [Kirchner and Klay, 1990, Nipkow, 1990]: An equational presentation E is said be to *resolvent*, if for any E-equality $s =_E t$ there exists an equational proof $s \longleftrightarrow_E^* t$ such that $\longleftrightarrow_E^*$ includes **at most one** equational step $\longleftrightarrow_E$ applied at the **root** position.
A theory is *syntactic* if it has a resolvent presentation.

Examples: A, C, AC are syntactic.

Motivation: If a theory is *syntactic*, then it admits a set of mutation rules transforming any unification problem in a sound and complete way.

Unification and Matching in Syntactic Theories

Fact. Any finite theory E with finitary E-unification is syntactic [Kirchner and Klay, 1990], where E is said to be *finite* if every equivalence class of $=_E$ has finitely many terms.

➽ A sound and complete unification procedure for syntactic theories, but **not necessarily terminating**

➽ A sound, complete and **terminating** matching procedure for finite syntactic theories (A, C, AC, ...)

➽ A sound, complete and **terminating** unification procedure for particular subclasses of syntactic theories:

- shallow theories [Comon et al., 1994],
- theories closed by paramodulation [Lynch and Morawska, 2002],
- theories with the Finite Variant Property [Eeralla et al., 2019].

Union of Theories

A problem is usually expressed modulo a union of theories, e.g., $\mathcal{E} = \mathcal{E}_1 \cup \mathcal{E}_2$ where $\mathcal{E}_i = A(*_i)$, $C(*_i)$, $AC(*_i)$, ...

Combination methods: solve the problem in a modular way by reusing the solvers known for individual theories $\mathcal{E}_1$ and $\mathcal{E}_2$

Existing combination methods for unions of disjoint theories:

- unification in arbitrary theories [Schmidt-Schauß, 1989, Baader and Schulz, 1996]
- matching in regular theories [Nipkow, 1991]
- matching in "regulo-linear" theories [Ringeissen, 1996]

Unions of theories sharing only constructors initiated in [Domenjoud et al., 1994, Baader and Tinelli, 2002]

Solving in a Union of Theories

1. Separate the input problem into pure sub-problems (via variable abstraction)
2. Solve the pure sub-problems by applying the respective solvers
3. Merge the solutions by taking care of the following problematic cases:
   - conflict of theories: a variable can be instantiated in several theories

     $$x = t_1 \ , \ x = t_2$$

   - compound cycle: a cycle between several theories

     $$x = t_1[y] \ , \ y = t_2[x]$$

     where t_1 and t_2 are (non-variable) pure terms in distinct theories.

Disjoint Union of Theories

- [Yelick, 1987]
  The problematic cases are trivially solved (no solution) in any union of **regular and collapse-free** disjoint theories
  - A theory $\mathcal{E}$ is *regular* if for any $l = r \in \mathcal{E}$, l and r have the same set of variables.
  - A theory $\mathcal{E}$ is *collapse-free* if there is no axiom $l = x \in \mathcal{E}$, where x is a variable.
- [Schmidt-Schauß, 1989, Baader and Schulz, 1996]
  Use unification with constant restriction to solve the problematic cases in any union of disjoint theories

Non-disjoint Union of Theories

In this talk: study non-disjoint unions $\mathcal{E} = \mathcal{E}_1 \cup \mathcal{E}_2$

- What happens when the individual theories $\mathcal{E}_1$ and $\mathcal{E}_2$ are two "conservative extensions" of a shared subtheory E
- What happens when the solvers known for $\mathcal{E}_1$ and $\mathcal{E}_2$ are built as "extensions" of a solver for E?
- What happens when $\mathcal{E}_1$ and $\mathcal{E}_2$ are syntactic theories?

Assumption:
For $i = 1, 2$, $\mathcal{E}_i = F_i \cup E$ where F_i is E-**constructed**, and the function symbols shared by F_1 and F_2 occur necessarily in E.

E-Constructed Theories: Examples

Let $E = AC(*)$

Exponentiation: $EX = \{e(e(X, Y), Z) = e(X, Y * Z)\}$

Homomorphism: $H = \{e(X * Y, Z) = e(X, Z) * e(Y, Z)\}$

Homomorphic Exponentiation: $EXH = EX \cup H$

$F = EX, H, EXH, \ldots$

Union of Theories $\mathcal{E}_1 \cup \mathcal{E}_2$ where $\mathcal{E}_i = F_i \cup E$ and F_i is obtained from F by renaming any function symbol f by f_i if f does not occur in E.

E-Constructed Term Rewrite Systems

Consider the left-to-right orientation of the Exponentiation:

Let $AC = AC(*)$, $R = \{\ e(e(X, Y), Z) \rightarrow e(X, Y * Z)\ \}$.

(R, AC) is an AC-constructed Term Rewrite System:

- (R, AC) is AC-convergent: existence and unicity of normal forms modulo AC,
- all the symbols in AC are constructors for R: for any rule $l \rightarrow r \in R$, l is not rooted by the AC-symbol $*$.

Union of E-Constructed Rewrite Systems

Questions addressed in this talk:

- What happens when the individual theories are E-constructed TRSs (sharing only symbols in E)? And syntactic?
- What happens when the unification procedures known for E-constructed TRSs are built in a hierarchical way as extensions of a E-unification procedure?

Use of E-Unification in E-Constructed Rewrite Systems

Property. *If (R, E) is E-constructed, then E-unification is sound and complete to solve $R \cup E$-unification problems built over symbols of E.*

➽ A crucial property to build an $R \cup E$-unification procedure in a hierarchical way as a combined procedure including

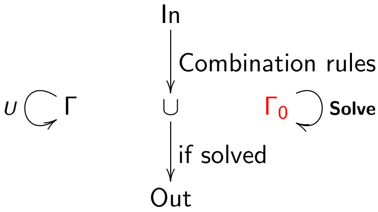
- an E-unification algorithm to solve all the equations that are *pure* in E,
- an additional inference system to solve all the other equations, typically via a set of mutation rules.

Hierarchical Unification Procedure

Let Σ_0 be the signature of E, and Σ the signature of $R \cup E$.

A hierarchical unification procedure $H_E(U)$ is given by:

- some combination rules, to get a *separate form* $\Gamma \cup \Gamma_0$ where Γ_0 is a set of Σ_0-equations and Γ is a set of $\Sigma \backslash \Sigma_0$-rooted flat equations.
- an E-unification algorithm (encapsulated into a **Solve** rule), to solve Γ_0
- an additional inference system U, to simplify Γ
  ➽ U may be a set of mutation rules, if $R \cup E$ is syntactic



Unification: Combination Rules

Coalesce $\{x = y\} \cup \Gamma \ \vdash \ \{x = y\} \cup (\Gamma\{x \mapsto y\})$
where x and y are distinct variables occurring both in Γ.

Split $\{f(\vec{v}) = t\} \cup \Gamma \ \vdash \ \{x = f(\vec{v}), x = t\} \cup \Gamma$
where $f \in \Sigma \backslash \Sigma_0$, t is a non-variable term and x is a fresh variable.

Flatten
$$\begin{array}{rl} & \{v = f(\ldots, u, \ldots)\} \cup \Gamma \\ \vdash & \{v = f(\ldots, x, \ldots), x = u\} \cup \Gamma \end{array}$$
where $f \in \Sigma \backslash \Sigma_0$, v is a variable, u is a non-variable term, and x is a fresh variable.

VA $\{s = t[u]\} \cup \Gamma \ \vdash \ \{s = t[x], x = u\} \cup \Gamma$
where t is Σ_0-rooted, u is an alien subterm of t, and x is a fresh variable.

Solving Rule

Solve $\Gamma \cup \Gamma_0 \ \vdash \ \bigvee_{\sigma_0 \in CSU_E(\Gamma_0)} \Gamma \cup \hat{\sigma}_0$

where

- Γ is a set of $\Sigma \backslash \Sigma_0$-equations,
- Γ_0 is a set of Σ_0-equations,
- Γ_0 is E-unifiable and not in tree solved form,
- $CSU_E(\Gamma_0)$ is a complete set of E-unifiers of Γ_0 **computed by an *E*-unification algorithm**,
- $\hat{\sigma}_0$ is the tree solved form associated to a unifier σ_0.

NB: **Solve** is implemented by calling an E-unification algorithm

Distributive Exponentiation

Let $AC = AC(\circledast)$

Consider two rewrite systems:

1. $$R_{\mathcal{E}} = \left\{ \begin{array}{l} exp(exp(X, Y), Z) \to exp(X, Y \circledast Z) \\ exp(X * Y, Z) \to exp(X, Z) * exp(Y, Z) \end{array} \right\}$$

2. $$R_{\mathcal{F}} = \{enc(enc(X, Y), Z) \to enc(X, Y \circledast Z)\}.$$

$(R_{\mathcal{E}}, AC)$ and $(R_{\mathcal{F}}, AC)$ are AC-constructed.

Unification in Distributive Exponentiation

Revisiting [Erbatur et al., 2011],

1. $\mathcal{E}_{AC} = R_{\mathcal{E}} \cup AC$ admits a hierarchical unification algorithm of the form $H_{AC}(U_{\mathcal{E}})$.
2. $\mathcal{F}_{AC} = R_{\mathcal{F}} \cup AC$ admits a hierarchical unification algorithm of the form $H_{AC}(U_{\mathcal{F}})$.

For instance, $U_{\mathcal{E}}$ includes the following rule:
$\{L = exp(v, w), L = exp(x, y)\} \cup \Gamma$
$\vdash \{L = exp(x, y), y = z \circledast w, v = exp(x, z)\} \cup \Gamma$.

Combined Hierarchical Unification

Let F_1 and F_2 be two E-constructed theories sharing only symbols in E such that

for $i = 1, 2$, $F_i \cup E$ admits a sound and complete unification procedure of the form $H_E(U_i)$.

Under which conditions do we that $H_E(U_1 \cup U_2)$ is a sound and complete unification procedure for $F_1 \cup F_2 \cup E$?

➽ Consider *layer-preserving* theories

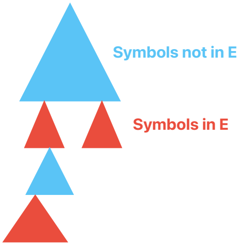
How to get a terminating $H_E(U_1 \cup U_2)$ procedure when $H_E(U_1)$ and $H_E(U_2)$ are both terminating?

➽ Consider a common decreasing measure

Layer-preserving Theories

Any term viewed as a "mounting" of two kinds of layers:

1. Σ_0-layer, built over Σ_0-symbols (the symbols in E),
2. $\Sigma \backslash \Sigma_0$-layer, built over $\Sigma \backslash \Sigma_0$-symbols.



An equational theory $F \cup E$ is said to be *layer-preserving* if, e.g., any term rooted by a $\Sigma \backslash \Sigma_0$-layer is necessarily equal modulo $F \cup E$ to a term rooted by a $\Sigma \backslash \Sigma_0$-layer.

Example: distributive exponentiation theories

Termination of Combined Hierarchical Unification

Find a complexity measure defined as a mapping C from separate forms to natural numbers such that $H_E(U)$ inference system is *C-decreasing*,

where C-decreasingness is a **modular property**:

If $H_E(U_1)$ and $H_E(U_2)$ are C-decreasing,
then $H_E(U_1 \cup U_2)$ is C-decreasing.

Union of Distributive Exponentiation Theories

Consider the distributive exponentiation theories $\mathcal{E}_{AC}$ and $\mathcal{F}_{AC}$ and their respective hierarchical unification algorithms $H_{AC}(U_{\mathcal{E}})$ and $H_{AC}(U_{\mathcal{F}})$.

- $\mathcal{E}_{AC}$ and $\mathcal{F}_{AC}$ are layer-preserving AC-constructed theories.

  Consequence: $H_{AC}(U_{\mathcal{E}} \cup U_{\mathcal{F}})$ is sound and complete.

- There exists a complexity measure SVC defined according to the number of equivalence classes of abstraction variables shared by Γ and Γ_0 such that:

  $H_{AC}(U_{\mathcal{E}} \cup U_{\mathcal{F}})$ is a SVC-decreasing
  since $H_{AC}(U_{\mathcal{E}})$ and $H_{AC}(U_{\mathcal{F}})$ are both SVC-decreasing.

  Consequence: $H_{AC}(U_{\mathcal{E}} \cup U_{\mathcal{F}})$ is also **terminating**.

Beyond E-Constructed TRSs

An equational theory F is *E-constructed* if there exists a normalizing mapping NF satisfying some properties including

$$s =_{F \cup E} t \text{ iff } NF(s) =_E NF(t)$$

and for any function symbol f in E,

$$NF(f(t_1, \ldots, t_n)) =_E f(NF(t_1), \ldots, NF(t_n))$$

Consequence: $F \cup E$-equality is decidable if NF is computable and E-equality is decidable.

Property: the class of E-constructed theories is closed by non-disjoint union (sharing only the symbols in E).

Remark: the definition of an E-constructed theory does not require that NF is computable.

E-Constructed Theories: More Examples

The following theories are E-constructed,
if E is the empty theory over $\{pk\}$:

$$K = \{keyex(X, pk(X'), Y, pk(Y')) = keyex(X', pk(X), Y', pk(Y))\}$$

$$ENC = \left\{ \begin{array}{rcl} adec(aenc(M, pk(S)), S) & = & M \\ checksign(sign(M, S), M, pk(S)) & = & ok \\ getmsg(sign(M, S)) & = & M \\ sdec(senc(M, K), K) & = & M \end{array} \right\}$$

From Unification to Matching

Consider any set of equations $\{\ldots\ ,\ s = t\ ,\ \ldots\}$

Unification problem: s and t are arbitrary.

Matching problem: s **or** t is ground.

Word problem: s **and** t are ground.

A key principle to solve Γ: eargerly normalize ground terms in Γ, via an appropriate normalizing mapping,

not necessarily NF, since NF is not assumed to be computable.

In practice, use of a weaker notion of normal form for any term t, called layer-reduced form of t, denoted by $t\Downarrow$, such that $t\Downarrow$ has the the same mounting of layers as $NF(t)$.

Combined Word Problem

Combination Theorem. Let F_1 and F_2 be any E-constructed theories sharing only symbols in E such that

for $i = 1, 2$, $F_i \cup E$ has a layer-reduced term mapping $\Downarrow_i$ and a decidable equality.

Then, $F_1 \cup F_2 \cup E$ has a (combined) layer-reduced term mapping $\Downarrow_{1,2}$ and a decidable equality.

Matching in Regular Theories

Property. In regular theories, any solution of any matching problem is necessarily ground: it is a matching problem in solved form.

Consider a matching problem $\{s_1[x] = t_1, s_2[x] = t_2\}$. For $i = 1, 2$, solving $s_i[x] = t_i$ yields $x = t'_i$ where t'_i is ground. Then, we just have to check whether $t'_1 = t'_2$.

Consequences for combined matching in regular theories:

- conflicts solved by checking (ground) equalities,
- no compound cycle.

Combined Matching in Regular Theories

Combination Theorem. Let F_1 and F_2 be any regular E-constructed theories sharing only symbols in E such that

for $i = 1, 2$, $F_i \cup E$ has a layer-reduced term mapping $\Downarrow_i$ and a matching algorithm.

Then, $F_1 \cup F_2 \cup E$ has a (combined) layer-reduced term mapping $\Downarrow_{1,2}$ and a (combined) matching algorithm.

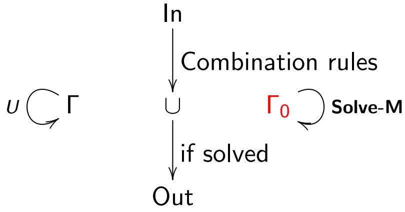
Question(s):

- What happens when the matching algorithms for $F_1 \cup E$ and $F_2 \cup E$ can be expressed in a hierarchical way?
- How to get a (combined) hierarchical matching algorithm for $F_1 \cup F_2 \cup E$?

Hierarchical Matching Procedure

A hierarchical $F \cup E$-matching procedure $HM_E(\Downarrow, U)$ given by:

- a layer-reduced term mapping $\Downarrow$,
- some fixed combination rules, to get a *separate form* $\Gamma \cup \Gamma_0$ such that Γ_0 (resp., Γ) is a set of match-equations where the non-ground terms are built over symbols in E (resp., symbols not in E),
- an E-matching algorithm **Solve-M** to solve Γ_0: can be applied **without loss of completeness**,
- an additional inference system U to simplify/mutate Γ.



Matching: Combination Rules

Let Σ_0 be the signature of E, and Σ the signature of $F \cup E$.

Norm $\{s = t\} \cup \Gamma \ \vdash\ \{s = t{\Downarrow}\} \cup \Gamma$
where t is ground and $t{\Downarrow} \neq t$.

Triv $\{s = t\} \cup \Gamma \ \vdash\ \Gamma$
where s, t are ground, $s{\Downarrow} = s$, $t{\Downarrow} = t$, and $s =_{F \cup E} t$.

Rep $\{x = t\} \cup \Gamma \ \vdash\ \{x = t\} \cup (\Gamma\{x \mapsto t\})$
where x is a variable occurring in Γ and t is a ground term.

Flatten-M $\{f(\vec{u}) = t\} \cup \Gamma \ \vdash\ \{f(\vec{x}) = t, \vec{u} = \vec{x}\} \cup \Gamma$
where $f(\vec{u})$ is a non-ground $\Sigma \backslash \Sigma_0$-rooted term, t is ground, and $\vec{x}$ are fresh variables.

VA-M $\{s[u] = t\} \cup \Gamma \ \vdash\ \{s[x] = t, u = x\} \cup \Gamma$
where s is a non-ground Σ_0-rooted term, u is an alien subterm of s, t is a ground, and x is a fresh variable.

Applying the Procedure

Let $F = \{h(X * Y) = h(X) * h(Y)\}$ and $E = AC(*)$.

$$
\begin{array}{ll}
\text{Input} & x * h(b) = h(a * b * c) \\
\vdash_{\textbf{Norm}} & x * h(b) = h(a) * h(b) * h(c) \\
\vdash_{\textbf{VA-M}} & x * v = h(a) * h(b) * h(c), v = h(b)
\end{array}
$$

1. $$
\begin{array}{ll}
\vdash_{\textbf{Solve-M}} & x = h(a), v = h(b) * h(c), v = h(b) \\
\vdash_{\textbf{Rep}} & x = h(a), v = h(b), h(b) = h(b) * h(c)
\end{array}
$$
2. $$
\begin{array}{ll}
\vdash_{\textbf{Solve-M}} & x = h(a) * h(b), v = h(c), v = h(b) \\
\vdash_{\textbf{Rep}} & x = h(a) * h(b), v = h(b), h(b) = h(c)
\end{array}
$$
3. $$
\begin{array}{ll}
\vdash_{\textbf{Solve-M}} & x = h(a) * h(c), v = h(b), v = h(b) \\
\vdash_{\textbf{Rep}} & x = h(a) * h(c), v = h(b), h(b) = h(b) \\
\vdash_{\textbf{Triv}} & x = h(a) * h(c), v = h(b)
\end{array}
$$
4. ...

Hierarchical Matching Algorithms: Examples

Let $F = \{h(X * Y) = h(X) * h(Y)\}$ and $E = AC(*)$.

$$h(x) = h(a) * h(b) \vdash_U x = a * b$$

The E-constructed TRS $(\{h(X * Y) \rightarrow h(X) * h(Y)\}, E)$ is an *innermost resolvent presentation* of $F \cup E$, where any innermost rewrite derivation has **at most one step applied at the root**.

Similar to the definition of resolvent presentation [Kirchner and Klay, 1990, Nipkow, 1990]

Result. If a theory has a (*innermost*) *resolvent* presentation, then it admits a set of mutation rules U leading to a sound and complete matching algorithm $HM_E(\Downarrow, U)$.

Combined Hierarchical Matching

Combination Theorem. Let F_1 and F_2 be any regular E-constructed theories sharing only symbols in E such that

for $i = 1, 2$, $F_i \cup E$ has a matching algorithm $HM_E(\Downarrow_i, U_i)$.

Then, $F_1 \cup F_2 \cup E$ has a matching algorithm $HM_E(\Downarrow_{1,2}, U_1 \cup U_2)$.

Combined Hierachical Solving

Development of a hierarchical solving framework dedicated to E-constructed theories.

- study of several terminating scenarios
  1. unification in forward-closed E-constructed TRSs [Erbatur et al., 2020]
  2. unification in paramodulation-closed E-constructed theories [Erbatur et al., 2021]
  3. matching in regular E-constructed theories (and word problem in arbitrary E-constructed theories) [Erbatur et al., 2022]
- Future work:
  - matching: beyond regular theories?
  - unification: a uniform treatment of terminating cases?
  - disunification?
  - knowledge problems arising in protocol analysis

References I

Baader, F. and Schulz, K. U. (1996).
Unification in the union of disjoint equational theories: Combining decision procedures.
J. Symb. Comput., 21(2):211–243.

Baader, F. and Tinelli, C. (2002).
Combining decision procedures for positive theories sharing constructors.
In Tison, S., editor, *Rewriting Techniques and Applications, 13th International Conference, RTA 2002, Copenhagen, Denmark, July 22-24, 2002, Proceedings*, volume 2378 of *Lecture Notes in Computer Science*, pages 352–366. Springer.

Comon, H., Haberstrau, M., and Jouannaud, J.-P. (1994).
Syntacticness, cycle-syntacticness, and shallow theories.
Inf. Comput., 111(1):154–191.

Domenjoud, E., Klay, F., and Ringeissen, C. (1994).
Combination techniques for non-disjoint equational theories.
In Bundy, A., editor, *Automated Deduction - CADE-12, 12th International Conference on Automated Deduction, Nancy, France, June 26 - July 1, 1994, Proceedings*, volume 814 of *Lecture Notes in Computer Science*, pages 267–281. Springer.

Eeralla, A. K., Erbatur, S., Marshall, A. M., and Ringeissen, C. (2019).
Rule-based unification in combined theories and the finite variant property.
In Martín-Vide, C., Okhotin, A., and Shapira, D., editors, *Language and Automata Theory and Applications - 13th International Conference, LATA 2019, St. Petersburg, Russia, March 26-29, 2019, Proceedings*, volume 11417 of *Lecture Notes in Computer Science*, pages 356–367. Springer.

Erbatur, S., Marshall, A. M., Kapur, D., and Narendran, P. (2011).
Unification over distributive exponentiation (sub)theories.
J. Autom. Lang. Comb., 16(2-4):109–140.

References II

Erbatur, S., Marshall, A. M., and Ringeissen, C. (2020).
Terminating non-disjoint combined unification.
In Fernández, M., editor, *Logic-Based Program Synthesis and Transformation - 30th International Symposium, LOPSTR 2020, Bologna, Italy, September 7-9, 2020, Proceedings*, volume 12561 of *Lecture Notes in Computer Science*, pages 113–130. Springer.

Erbatur, S., Marshall, A. M., and Ringeissen, C. (2021).
Non-disjoint combined unification and closure by equational paramodulation.
In Konev, B. and Reger, G., editors, *Frontiers of Combining Systems*, pages 25–42, Cham. Springer International Publishing.

Erbatur, S., Marshall, A. M., and Ringeissen, C. (2022).
Combined hierarchical matching: The regular case.
In Felty, A. P., editor, *7th International Conference on Formal Structures for Computation and Deduction, FSCD 2022, August 2-5, 2022, Haifa, Israel*, volume 228 of *LIPIcs*, pages 26:1–26:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

Kirchner, C. and Klay, F. (1990).
Syntactic theories and unification.
In *Proceedings of the Fifth Annual Symposium on Logic in Computer Science (LICS '90), Philadelphia, Pennsylvania, USA, June 4-7, 1990*, pages 270–277. IEEE Computer Society.

Lynch, C. and Morawska, B. (2002).
Basic syntactic mutation.
In Voronkov, A., editor, *Automated Deduction - CADE-18, 18th International Conference on Automated Deduction, Copenhagen, Denmark, July 27-30, 2002, Proceedings*, volume 2392 of *Lecture Notes in Computer Science*, pages 471–485. Springer.

References III

Nipkow, T. (1990).
Proof transformations for equational theories.
In *Proceedings of the Fifth Annual Symposium on Logic in Computer Science (LICS '90), Philadelphia, Pennsylvania, USA, June 4-7, 1990*, pages 278–288. IEEE Computer Society.

Nipkow, T. (1991).
Combining matching algorithms: The regular case.
J. Symb. Comput., 12(6):633–654.

Ringeissen, C. (1996).
Combining decision algorithms for matching in the union of disjoint equational theories.
Inf. Comput., 126(2):144–160.

Schmidt-Schauß, M. (1989).
Unification in a combination of arbitrary disjoint equational theories.
J. Symb. Comput., 8(1/2):51–99.

Yelick, K. A. (1987).
Unification in combinations of collapse-free regular theories.
Journal of Symbolic Computation, 3(1-2):153–181.