Order-Sorted Unification with Regular Expression Sorts
(Work in Progress)

Temur Kutsia\textsuperscript{1}  Mircea Marin\textsuperscript{2}

\textsuperscript{1}RISC, JKU Linz, Austria.
\textsuperscript{2}University of Tsukuba, Japan

Presented by Laura Kovács
Advantages of using sorts in logic:

- More adequate coding of mathematical problems in logic.
- Simplification of algebraic specifications.
- Compact and natural formalizations of AI problems.
- Supporting typed computations.
- ...
Motivation

Sorted unification:

- Walther in 1988 extended first-order unification to order-sorted case:
  - Infinitely many sorts, ordered
  - No overloading
  - Problems may have infinitely many unifiers
  - Complete procedure to enumerate unifiers
  - Minimal complete set of unifiers might not exist
  - If the set of sorts is finite, the problems becomes finitary

Further generalizations: Schmidt-Schauss 1989, Weidenbach 1996:

- Sorts as unary predicates
- Term declarations allowed

Undecidable

Other works: Kirchner 1988, Hendrix, Mesegeur 2008 (equational OSU).
Motivation

Sorted unification:

- Walther in 1988 extended first-order unification to order-sorted case:
  - Infinitely many sorts, ordered
  - No overloading
  - Problems may have infinitely many unifiers
  - Complete procedure to enumerate unifiers
  - Minimal complete set of unifiers might not exist
  - If the set of sorts is finite, the problems becomes finitary

- Further generalizations: Schmidt-Schauss 1989, Weidenbach 1996:
  - Sorts as unary predicates
  - Term declarations allowed
  - Undecidable
Motivation

Sorted unification:

- Walther in 1988 extended first-order unification to order-sorted case:
  - Infinitely many sorts, ordered
  - No overloading
  - Problems may have infinitely many unifiers
  - Complete procedure to enumerate unifiers
  - Minimal complete set of unifiers might not exist
  - If the set of sorts is finite, the problems becomes finitary

- Further generalizations: Schmidt-Schauss 1989, Weidenbach 1996:
  - Sorts as unary predicates
  - Term declarations allowed
  - Undecidable

- Other works: Kirchner 1988, Hendrix, Mesegeur 2008 (equational OSU).
Motivation

Closer look at Walther’s work:

- Set of sorts $S$ and order $\leq$.
- Set of variables $\mathcal{V}_s$ for each $s \in S$.
- Set of function symbols $\mathcal{F}_{w,s}$ for each $w \in S^*, s \in S$.
- All these sets are pairwise disjoint.
Motivation

Closer look at Walther’s work:

- Set of sorts $S$ and order $\leq$.
- Set of variables $\mathcal{V}_s$ for each $s \in S$.
- Set of function symbols $\mathcal{F}_{w,s}$ for each $w \in S^*, s \in S$.
- All these sets are pairwise disjoint.

Our contribution:

- Finite set of basic sorts $B$ and order $\preceq$.
- Sorts are regular expressions over $B$, denoted $R\preceq$ extends to sorts.
- Set of variables $\mathcal{V}_R$ for each $R \in R$.
- Set of function symbols $\mathcal{F}_R,s$ for each $R \in R$, $s \in B$.
- Sets of function symbols are not required to be disjoint.
Motivation

Closer look at Walther’s work:

- Set of sorts $S$ and order $\leq$.
- Set of variables $\mathcal{V}_s$ for each $s \in S$.
- Set of function symbols $\mathcal{F}_{w,s}$ for each $w \in S^*, s \in S$.
- All these sets are pairwise disjoint.

Our contribution:

- Finite set of basic sorts $\mathcal{B}$ and order $\preceq$. 


Motivation

Closer look at Walther’s work:
- Set of sorts $S$ and order $\leq$.
- Set of variables $\mathcal{V}_s$ for each $s \in S$.
- Set of function symbols $\mathcal{F}_{w,s}$ for each $w \in S^*$, $s \in S$.
- All these sets are pairwise disjoint.

Our contribution:
- Finite set of basic sorts $B$ and order $\preceq$.
- Sorts are regular expressions over $B$, denoted $\mathcal{R}$. $\preceq$ extends to sorts.
Motivation

Closer look at Walther’s work:

- Set of sorts $S$ and order $\leq$.
- Set of variables $V_s$ for each $s \in S$.
- Set of function symbols $F_{w,s}$ for each $w \in S^*, s \in S$.
- All these sets are pairwise disjoint.

Our contribution:

- Finite set of basic sorts $B$ and order $\preceq$.
- Sorts are regular expressions over $B$, denoted $\mathcal{R}$. $\preceq$ extends to sorts.
- Set of variables $V_R$ for each $R \in \mathcal{R}$.
Closer look at Walther’s work:

- Set of sorts $S$ and order $\leq$.
- Set of variables $\mathcal{V}_s$ for each $s \in S$.
- Set of function symbols $\mathcal{F}_{w,s}$ for each $w \in S^*, s \in S$.
- All these sets are pairwise disjoint.

Our contribution:

- Finite set of basic sorts $\mathcal{B}$ and order $\preceq$.
- Sorts are regular expressions over $\mathcal{B}$, denoted $\mathcal{R}$. $\preceq$ extends to sorts.
- Set of variables $\mathcal{V}_R$ for each $R \in \mathcal{R}$.
- Set of function symbols $\mathcal{F}_{R,s}$ for each $R \in \mathcal{R}, s \in \mathcal{B}$.
Motivation

Closer look at Walther’s work:

- Set of sorts $S$ and order $\leq$.
- Set of variables $\mathcal{V}_s$ for each $s \in S$.
- Set of function symbols $\mathcal{F}_{w,s}$ for each $w \in S^*, s \in S$.
- All these sets are pairwise disjoint.

Our contribution:

- Finite set of basic sorts $\mathcal{B}$ and order $\preceq$.
- Sorts are regular expressions over $\mathcal{B}$, denoted $\mathcal{R}$. $\preceq$ extends to sorts.
- Set of variables $\mathcal{V}_R$ for each $R \in \mathcal{R}$.
- Set of function symbols $\mathcal{F}_{R,s}$ for each $R \in \mathcal{R}, s \in \mathcal{B}$.
- Sets of function symbols are not required to be disjoint.
Related Work

- SYNU - Syntactic unification (Robinson 1965).
- WU - Word unification (Schulz 1990).
- OSU - Order-sorted unification (Walther 1988).
- SEQU - Sequence unification (Kutsia 2002).
- WURC - Word unification with regular constraints (Schulz 1990).
- REOSU - Regular expression order-sorted unification.
Example

- Solve \( f(x, y, z) = f(f(x), g(u), a, b) \)

- Sort information.
  Basic sorts: \( s, r, q \). Ordering: \( s \prec q, r \prec q \).

\[
\begin{align*}
x, z & : s^* \\
y, u & : q \\
a, b & : s \\
f & : q^* \rightarrow r \\
g & : q \rightarrow q, s + r \rightarrow s
\end{align*}
\]

- Solution: \( \{x \mapsto \epsilon, y \mapsto f(\epsilon), u \mapsto v, z \mapsto (g(v), a, b)\} \).

- \( v \): fresh variable of sort \( s + r \).

- \( u \mapsto v \) weakens \( g(u) : q \) to \( g(v) : s \).
Sorts

- Finite set $B$ of basic sorts, partially ordered with the relation $\preceq$.
- $s, r, q$ denote basic sorts.
- Regular expression sorts:
  \[
  R ::= s \mid 1 \mid R_1.R_2 \mid R_1+R_2 \mid R^*.
  \]
- Corresponding regular language $[[R]]$: The set of words of base sorts, defined as usual.
Sorts

Extensions of $\preceq$:

- On words of base sorts:
  
  $s_1 \cdots s_n \preceq r_1 \cdots r_n$ iff $s_i \preceq r_i$ for all $1 \leq i \leq n$.

- On sets of words of base sorts:
  
  $S_1 \preceq S_2$ iff for each $w_1 \in S_1$ there is $w_2 \in S_2$ such that $w_1 \preceq w_2$.

- On (regular expression) sorts:
  
  $R_1 \preceq R_2$ iff $[R_1] \preceq [R_2]$.

- $R_1 \simeq R_2$ means $R_1 \preceq R_2$ and $R_2 \preceq R_1$. 
Extensions of $\preceq$:

- On words of base sorts:
  \[ s_1 \cdots s_n \preceq r_1 \cdots r_n \text{ iff } s_i \preceq r_i \text{ for all } 1 \leq i \leq n. \]

- On sets of words of base sorts:
  \[ S_1 \preceq S_2 \text{ iff for each } w_1 \in S_1 \text{ there is } w_2 \in S_2 \text{ such that } w_1 \preceq w_2. \]
Sorts

Extensions of $\preceq$:

- On words of base sorts:
  \[ s_1 \cdots s_n \preceq r_1 \cdots r_n \text{ iff } s_i \preceq r_i \text{ for all } 1 \leq i \leq n. \]

- On sets of words of base sorts:
  \[ S_1 \preceq S_2 \text{ iff for each } w_1 \in S_1 \text{ there is } w_2 \in S_2 \text{ such that } w_1 \preceq w_2. \]

- On (regular expression) sorts:
  \[ R_1 \preceq R_2 \text{ iff } \llbracket R_1 \rrbracket \preceq \llbracket R_2 \rrbracket. \]
Extensions of $\preceq$:

- On words of base sorts:
  \[ s_1 \cdots s_n \preceq r_1 \cdots r_n \iff s_i \preceq r_i \text{ for all } 1 \leq i \leq n. \]

- On sets of words of base sorts:
  \[ S_1 \preceq S_2 \iff \text{for each } w_1 \in S_1 \text{ there is } w_2 \in S_2 \text{ such that } w_1 \preceq w_2. \]

- On (regular expression) sorts:
  \[ R_1 \preceq R_2 \iff \llbracket R_1 \rrbracket \preceq \llbracket R_2 \rrbracket. \]

- \( R_1 \simeq R_2 \) means \( R_1 \preceq R_2 \) and \( R_2 \preceq R_1 \).
Alphabet

- Countable set of variables $\mathcal{V}_R$ for each sort $R$, satisfying conditions:
  - $\mathcal{V}_{R_1} = \mathcal{V}_{R_2}$ if $R_1 \simeq R_2$.
  - $\mathcal{V}_{R_1} \cap \mathcal{V}_{R_2} = \emptyset$ if $R_1 \not\simeq R_2$. 
Alphabet

- Countable set of variables $\mathcal{V}_R$ for each sort $R$, satisfying conditions:
  - $\mathcal{V}_{R_1} = \mathcal{V}_{R_2}$ if $R_1 \simeq R_2$.
  - $\mathcal{V}_{R_1} \cap \mathcal{V}_{R_2} = \emptyset$ if $R_1 \not\simeq R_2$.

- A family of sets of function symbols $\{\mathcal{F}_{R,s} \mid R \in \mathcal{R}, s \in \mathcal{B}\}$, satisfying conditions:
  - $\mathcal{F}_{R_1,s_1} = \mathcal{F}_{R_2,s_2}$ iff $R_1.s_1 \simeq R_2.s_2$
  - Monotonicity: If $f \in \mathcal{F}_{R_1,s_1} \cap \mathcal{F}_{R_2,s_2}$ and $R_1 \preceq R_2$, then $s_1 \preceq s_2$.
  - Preregularity: If $f \in \mathcal{F}_{R_1,s_1}$ and $R_2 \preceq R_1$, then there is a $\preceq$-least element in the set $\{s \mid f \in \mathcal{F}_{R,s} \text{ and } R_2 \preceq R\}$.
  - Finite overloading: For each $f$, the set $\{\mathcal{F}_{R,s} \mid f \in \mathcal{F}_{R,s}\}$ is finite.
Alphabet

- Countable set of variables $\mathcal{V}_R$ for each sort $R$, satisfying conditions:
  - $\mathcal{V}_{R_1} = \mathcal{V}_{R_2}$ if $R_1 \simeq R_2$.
  - $\mathcal{V}_{R_1} \cap \mathcal{V}_{R_2} = \emptyset$ if $R_1 \not\simeq R_2$.
- A family of sets of function symbols $\{\mathcal{F}_{R,s} \mid R \in \mathcal{R}, s \in \mathcal{B}\}$, satisfying conditions:
  - $\mathcal{F}_{R_1,s_1} = \mathcal{F}_{R_2,s_2}$ iff $R_1.s_1 \simeq R_2.s_2$
  - Monotonicity: If $f \in \mathcal{F}_{R_1,s_1} \cap \mathcal{F}_{R_2,s_2}$ and $R_1 \preceq R_2$, then $s_1 \preceq s_2$.
  - Preregularity: If $f \in \mathcal{F}_{R_1,s_1}$ and $R_2 \preceq R_1$, then there is a $\preceq$-least element in the set $\{s \mid f \in \mathcal{F}_{R,s} \text{ and } R_2 \preceq R\}$.
  - Finite overloading: For each $f$, the set $\{\mathcal{F}_{R,s} \mid f \in \mathcal{F}_{R,s}\}$ is finite.

- Notation. We use
  - $f : R \rightarrow s$ for $f \in \mathcal{F}_{R,s}$
  - $a : s$ for $a \in \mathcal{F}_{1,s}$
  - $x : R$ for $x \in \mathcal{V}_R$
The set of terms over $\mathcal{V} = \bigcup_{R \in \mathcal{R}} \mathcal{V}_R$ and $\mathcal{F} = \bigcup_{R \in \mathcal{R}, s \in \mathcal{B}} \mathcal{F}_{R,s}$: The least set $\mathcal{I}(\mathcal{F}, \mathcal{V}) = \{ \mathcal{I}_R(\mathcal{F}, \mathcal{V}) \mid R \in \mathcal{R} \}$ such that

- $\mathcal{V}_R \subseteq \mathcal{I}_R(\mathcal{F}, \mathcal{V})$.
- $\mathcal{I}_{R_1}(\mathcal{F}, \mathcal{V}) \subseteq \mathcal{I}_{R_2}(\mathcal{F}, \mathcal{V})$ if $R_1 \preceq R_2$.
- If $f : R \to s$ and $1 \preceq R$, then $f(\epsilon) \in \mathcal{I}_s(\mathcal{F}, \mathcal{V})$.
- If $f : R \to s$, $t_i \in \mathcal{I}_{R_i}(\mathcal{F}, \mathcal{V})$ for $1 \leq i \leq n$, $n \geq 1$, such that $R_1.\ldots.R_n \preceq R$, then $f(t_1,\ldots,t_n) \in \mathcal{I}_s(\mathcal{F}, \mathcal{V})$. 
Terms

The set of terms over $\mathcal{V} = \bigcup_{R \in \mathcal{R}} \mathcal{V}_R$ and $\mathcal{F} = \bigcup_{R \in \mathcal{R}, s \in \mathcal{B}} \mathcal{F}_{R,s}$: The least set $\mathcal{T}(\mathcal{F}, \mathcal{V}) = \{ \mathcal{T}_R(\mathcal{F}, \mathcal{V}) \mid R \in \mathcal{R} \}$ such that

- $\mathcal{V}_R \subseteq \mathcal{T}_R(\mathcal{F}, \mathcal{V})$.
- $\mathcal{T}_{R_1}(\mathcal{F}, \mathcal{V}) \subseteq \mathcal{T}_{R_2}(\mathcal{F}, \mathcal{V})$ if $R_1 \preceq R_2$.
- If $f : R \to s$ and $1 \preceq R$, then $f(\epsilon) \in \mathcal{T}_s(\mathcal{F}, \mathcal{V})$.
- If $f : R \to s$, $t_i \in \mathcal{T}_{R_i}(\mathcal{F}, \mathcal{V})$ for $1 \leq i \leq n$, $n \geq 1$, such that $R_1 \ldots R_n \preceq R$, then $f(t_1, \ldots, t_n) \in \mathcal{T}_s(\mathcal{F}, \mathcal{V})$.

Lemma

For each term $t$ there exists a $\preceq$-minimal sort $R$ that is unique modulo $\simeq$ such that $t \in \mathcal{T}_R(\mathcal{F}, \mathcal{V})$. 
Semantics of Sorts

- $\text{sem}(s) = \mathcal{T}_s(\mathcal{F})$: Semantics of a basic sort $s$ is the set of ground terms of sort $s$.
- Semantics of a regular sort is the set of ground term sequences of the corresponding sort:
  - $\text{sem}(1) = \{\epsilon\}$.
  - $\text{sem}(R_1.R_2) = \{ (\tilde{s}_1, \tilde{s}_2) \mid \tilde{s}_1 \in \text{sem}(R_1), \tilde{s}_2 \in \text{sem}(R_2) \}$.
  - $\text{sem}(R_1 + R_2) = \text{sem}(R_1) \cup \text{sem}(R_2)$.
  - $\text{sem}(R^*) = \text{sem}(R)^*$.
Substitutions

- Substitution: a well-sorted mapping from variables to sequences of terms, which is identity almost everywhere.
- Well-sortedness (of \(\sigma\)): \(\text{sort}(\sigma(x)) \preceq \text{sort}(x)\) for all \(x\).
- Standard notions defined in the usual way.
- Subsumption \(\sigma \preceq_x \vartheta\): \(\sigma\) is more general than \(\vartheta\) on the set of variables \(x\).
Algorithms for Sorts

We will need to

- Decide $\preceq$.
- Compute greatest lower bounds for regular expressions.
- Compute weakening substitutions.

N.B. Our basic sorts are ordered.
Algorithms for Sorts. Deciding \( \subseteq \)

- **closure** \( \overline{R} \) of a sort \( R \):
  - \( \overline{s} = \sum_{r \preceq s} r \)
  - \( 1 = 1 \)
  - \( \overline{R_1.R_2} = \overline{R_1}.\overline{R_2} \)
  - \( \overline{R_1+R_2} = \overline{R_1}+\overline{R_2} \)
  - \( \overline{R^*} = \overline{R}^* \)

- Nice property: \( S \preceq R \) iff \( \llbracket S \rrbracket \subseteq \llbracket R \rrbracket \).
- Deciding \( \llbracket S \rrbracket \subseteq \llbracket R \rrbracket \) does not need to take into account ordering on basic sorts.
- \( \llbracket S \rrbracket \subseteq \llbracket R \rrbracket \) can be decided by Antimirov’s algorithm.
- The problem is PSPACE-complete.
Without ordering on basic sorts, glb of regular expression sorts is their intersection.

In our case, we modify Antimirov-Mosses intersection algorithm:

- To intersect two basic sorts, compute maximal elements in the set of their lower bounds.
- The number of such maximal elements is finite. Their sum is the glb of the given basic sorts.

Requires double-exponential time.
Algorithms for Sorts. Compute Weakening Substitutions

- Weakening substitution for a term sequence $\tilde{t}$ towards a sort $R$: A substitution $\sigma$ such that $\text{sort}(\tilde{t}\sigma) \preceq R$.
- What is it good for?

Example

- Let $x : s$, $f : R_1 \rightarrow s_1$, $f : R_2 \rightarrow s_2$, $y : R_2$.
- Sort ordering: $R_1 \preceq R_2$, $s_1 \preceq s \preceq s_2$. 
Weakening substitution for a term sequence $\tilde{t}$ towards a sort $R$: A substitution $\sigma$ such that $\text{sort}(\tilde{t}\sigma) \preceq R$.

What is it good for?

Example

Let $x : s$, $f : R_1 \rightarrow s_1$, $f : R_2 \rightarrow s_2$, $y : R_2$.

Sort ordering: $R_1 \preceq R_2$, $s_1 \preceq s \preceq s_2$.

Then $x$ and $f(y)$ do not unify: $\text{sort}(f(y)) = s_2 \not\preceq s = \text{sort}(x)$. 

But, if we weaken $\text{sort}(f(y))$ to $s_1$, then unification is possible. To weaken, we take the instance of $f(y)$ under the substitution $\{ y \mapsto z \}$, where $\text{sort}(z) = R_1$, which gives $\text{sort}(f(z)) = s_1$. 

Algorithms for Sorts. Compute Weakening Substitutions
Algorithms for Sorts. Compute Weakening Substitutions

- Weakening substitution for a term sequence $\tilde{t}$ towards a sort $R$: A substitution $\sigma$ such that $\text{sort}(\tilde{t}\sigma) \preceq R$.
- What is it good for?

Example

- Let $x : s$, $f : R_1 \rightarrow s_1$, $f : R_2 \rightarrow s_2$, $y : R_2$.
- Sort ordering: $R_1 \preceq R_2$, $s_1 \preceq s \preceq s_2$.
- Then $x$ and $f(y)$ do not unify: $\text{sort}(f(y)) = s_2 \not\preceq s = \text{sort}(x)$.
- But, if we weaken $\text{sort}(f(y))$ to $s_1$, then unification is possible.
Weakening substitution for a term sequence $\tilde{t}$ towards a sort $R$: A substitution $\sigma$ such that $\text{sort}(\tilde{t}\sigma) \preceq R$.

What is it good for?

Example

Let $x : s$, $f : R_1 \rightarrow s_1$, $f : R_2 \rightarrow s_2$, $y : R_2$.

Sort ordering: $R_1 \preceq R_2$, $s_1 \preceq s \preceq s_2$.

Then $x$ and $f(y)$ do not unify: $\text{sort}(f(y)) = s_2 \not\preceq s = \text{sort}(x)$.

But, if we weaken $\text{sort}(f(y))$ to $s_1$, then unification is possible.

To weaken, we take the instance of $f(y)$ under the substitution $\{y \mapsto z\}$, where $\text{sort}(z) = R_1$, which gives $\text{sort}(f(z)) = s_1$. 
A rule-based algorithm has been developed to compute weakening substitutions.

The algorithm is sound, complete, and terminating.

The finite overloading property of the alphabet and the finite factorization property of regular expressions are important for termination.

Each weakening substitution is a variable renaming substitution.

Details in the paper.
Unification Type, Decidability, Procedure

- Type: Infinitary.
- Decidability: Unknown. (Conjectured to be decidable. No formal proof yet).
Unification Procedure.

Example

- Solve \( f(x, y, z) \equiv f(f(x), g(u), a, b) \)
- Basic sorts: \( s, r, q \). Ordering: \( s \prec q, r \prec q \).

\[
\begin{align*}
x, z : s^* & \quad f : q^* \rightarrow r \\
y, u : q & \quad g : q \rightarrow q, \ s + r \rightarrow s \\
a, b : s
\end{align*}
\]
Unification Procedure.

Example

- Solve $f(x, y, z) \doteq f(f(x), g(u), a, b)$
- Basic sorts: $s, r, q$. Ordering: $s \prec q, r \prec q.$

\[
\begin{array}{c}
x, z : s^* \quad f : q^* \rightarrow r \\
y, u : q \quad g : q \rightarrow q, \quad s + r \rightarrow s \\
a, b : s
\end{array}
\]

Unification:

\[
\{f(x, y, z) \doteq f(f(x), g(u), a, b)\}; \varepsilon \rightarrow_{decompose}
\]
Unification Procedure.

Example

- Solve \( f(x, y, z) \overset{\equiv}{=} f(f(x), g(u), a, b) \)
- Basic sorts: \( s, r, q \). Ordering: \( s \prec q, r \prec q \).

\[
\begin{align*}
  x, z & : s^* & f & : q^* \rightarrow r \\
  y, u & : q & g & : q \rightarrow q, \ s + r \rightarrow s \\
  a, b & : s
\end{align*}
\]

Unification:

\[
\{ f(x, y, z) \overset{\equiv}{=} f(f(x), g(u), a, b) \}; \varepsilon \overset{\text{decompose}}{\Longrightarrow} \\
\{ (x, y, z) \overset{\equiv}{=} (f(x), g(u), a, b) \}; \varepsilon \overset{\text{project}}{\Longrightarrow}
\]

Unification Procedure.

Example

- Solve \( f(x, y, z) \doteq f(f(x), g(u), a, b) \)
- Basic sorts: \( s, r, q \). Ordering: \( s \prec q, r \prec q \).

\[
\begin{align*}
x, z &: s^* \\
f &: q^* \rightarrow r \\
y, u &: q \\
g &: q \rightarrow q, \ s + r \rightarrow s \\
a, b &: s
\end{align*}
\]

Unification:

\[
\begin{align*}
\{ f(x, y, z) &\doteq f(f(x), g(u), a, b) \}; \varepsilon &\longrightarrow_{\text{decompose}} \\
\{ (x, y, z) &\doteq (f(x), g(u), a, b) \}; \varepsilon &\longrightarrow_{\text{project}} \\
\{ (y, z) &\doteq (f(e), g(u), a, b) \}; \{ x \mapsto e \} &\longrightarrow_{\text{eliminate}}
\end{align*}
\]
Unification Procedure.

Example

- Solve $f(x, y, z) \doteq f(f(x), g(u), a, b)$
- Basic sorts: $s, r, q$. Ordering: $s \prec q$, $r \prec q$.

\[
\begin{align*}
    x, z &: s^* & f &: q^* \to r \\
    y, u &: q & g &: q \to q, s + r &: s \\
    a, b &: s
\end{align*}
\]

Unification:

\[
\begin{align*}
    \{ z \doteq (g(u), a, b) \}; \{ x \mapsto \epsilon, y \mapsto f(\epsilon) \} \\
    \implies_{\text{weaken\_widen}} (\text{sort}(v) = s + r), (\text{sort}(z_1) = s^*)
\end{align*}
\]
Unification Procedure.

Example

▸ Solve \( f(x, y, z) \doteq f(f(x), g(u), a, b) \)
▸ Basic sorts: s, r, q. Ordering: \( s \prec q, r \prec q \).

\[
\begin{align*}
x, z & : s^* & f & : q^* \rightarrow r \\
y, u & : q & g & : q \rightarrow q, \quad s + r \rightarrow s \\
a, b & : s
\end{align*}
\]

Unification:

\[
\begin{align*}
\{ z \doteq (g(u), a, b) \}; \{ x \mapsto \epsilon, y \mapsto f(\epsilon) \} \\
\quad \xrightarrow{\text{weaken\_widen}} (\text{sort}(v) = s + r), (\text{sort}(z_1) = s^*) \\
\{ z_1 \doteq (a, b) \}; \{ x \mapsto \epsilon, y \mapsto f(\epsilon), u \mapsto v, z \mapsto (g(v), z_1) \} \\
\quad \xrightarrow{\text{widen}} (\text{sort}(z_2) = s^*)
\end{align*}
\]
Unification Procedure.

Example

- Solve $f(x, y, z) \doteq f(f(x), g(u), a, b)$
- Basic sorts: $s, r, q$. Ordering: $s \prec q, r \prec q$.

  $x, z : s^*$  \hspace{1cm} $f : q^* \rightarrow r$

  $y, u : q$  \hspace{1cm} $g : q \rightarrow q, s + r \rightarrow s$

  $a, b : s$

Unification:

\[
\{ z_2 \doteq b \}; \{ x \mapsto \epsilon, y \mapsto f(\epsilon), u \mapsto v, z \mapsto (g(v), a, z_2), z_1 \mapsto (a, z_2) \}\]

$\implies \text{eliminate}$
Unification Procedure.

Example

▶ Solve $f(x, y, z) \doteq f(f(x), g(u), a, b)$

▶ Basic sorts: $s, r, q$. Ordering: $s \prec q, r \prec q$.

$x, z : s^* 
\quad f : q^* \rightarrow r$

$y, u : q 
\quad g : q \rightarrow q, s + r \rightarrow s$

$a, b : s$

Unification:

\{z_2 \doteq b\}; \{x \mapsto \epsilon, y \mapsto f(\epsilon), u \mapsto v, z \mapsto (g(v), a, z_2), z_1 \mapsto (a, z_2)\}

\quad \Longrightarrow \text{eliminate}

\{\epsilon \doteq \epsilon\}; \{x \mapsto \epsilon, y \mapsto f(\epsilon), u \mapsto v, z \mapsto (g(v), a, b), z_1 \mapsto (a, b), z_2 \mapsto b\}

\quad \Longrightarrow \text{trivial}
Unification Procedure.

Example

- Solve $f(x, y, z) = f(f(x), g(u), a, b)$
- Basic sorts: $s, r, q$. Ordering: $s \prec q, r \prec q$.

$$x, z : s^* \quad f : q^* \rightarrow r$$
$$y, u : q \quad g : q \rightarrow q, \ s + r \rightarrow s$$
$$a, b : s$$

Unification:

$$\{z_2 \doteq b\}; \{x \mapsto \epsilon, y \mapsto f(\epsilon), u \mapsto v, z \mapsto (g(v), a, z_2), z_1 \mapsto (a, z_2)\}$$

$\implies_{\text{eliminate}}$

$$\{\epsilon \doteq \epsilon\}; \{x \mapsto \epsilon, y \mapsto f(\epsilon), u \mapsto v, z \mapsto (g(v), a, b), z_1 \mapsto (a, b), z_2 \mapsto b\}$$

$\implies_{\text{trivial}}$

$$\emptyset; \{x \mapsto \epsilon, y \mapsto f(\epsilon), u \mapsto v, z \mapsto (g(v), a, b), z_1 \mapsto (a, b), z_2 \mapsto b\}$$
Unification Procedure.

Example

- Solve \( f(x, y, z) = f(f(x), g(u), a, b) \)
- Basic sorts: \( s, r, q \). Ordering: \( s \prec q, r \prec q \).

\[
\begin{align*}
x, z : s^* & \quad f : q^* \rightarrow r \\
y, u : q & \quad g : q \rightarrow q, \ s + r \rightarrow s \\
a, b : s
\end{align*}
\]

Unification: (Restricting the solution to the original problem variables)

\[
\{x \mapsto \epsilon, y \mapsto f(\epsilon), u \mapsto v, z \mapsto (g(v), a, b)\}
\]

\(\text{sort}(v) = s + r\)
Summary

- Order-sorted unification extended to regular expression sorts.
- The obtained problem generalizes some known unification problems.
- Sort weakening algorithm constructed. Its soundness, completeness, and termination are proved.
- An algorithm for computing greatest lower bound of a set of sorts is sketched.
- Unification type is infinitary: proved.
- Procedure for enumerating a (minimal) complete set of unifiers constructed. Its soundness and completeness proved.
- Decidability to be proved.