Unification

3.7.1 Definition (Unifier)

Two terms *s* and *t* of the same sort are said to be *unifiable* if there exists a well-sorted substitution σ so that $s\sigma = t\sigma$, the substitution σ is then called a well-sorted *unifier* of *s* and *t*.

The unifier σ is called *most general unifier*, written $\sigma = mgu(s, t)$, if any other well-sorted unifier τ of *s* and *t* it can be represented as $\tau = \sigma\tau'$, for some well-sorted substitution τ'.

A state of the naive standard unification calculus is a set of equations E or $\bot$, where $\bot$ denotes that no unifier exists. The set E is also called a *unification problem*.

The start state for checking whether two terms s, t, $\mathrm{sort}(s) = \mathrm{sort}(t)$, (or two non-equational atoms A, B) are unifiable is the set $E = \{s = t\}$ ($E = \{A = B\}$). A variable x is *solved* in E if $E = \{x = t\} \uplus E'$, $x \notin \mathrm{vars}(t)$ and $x \notin \mathrm{vars}(E)$.

A variable $x \in \mathrm{vars}(E)$ is called *solved* in E if $E = E' \uplus \{x = t\}$ and $x \notin \mathrm{vars}(t)$ and $x \notin \mathrm{vars}(E')$.

Standard (naive) Unification

Tautology $E \uplus \{t = t\} \;\Rightarrow_{\mathrm{SU}}\; E$

Decomposition $E \uplus \{f(s_1, \ldots, s_n) = f(t_1, \ldots, t_n)\} \;\Rightarrow_{\mathrm{SU}}$
$E \cup \{s_1 = t_1, \ldots, s_n = t_n\}$

Clash $E \uplus \{f(s_1, \ldots, s_n) = g(s_1, \ldots, s_m)\} \;\Rightarrow_{\mathrm{SU}}\; \perp$
if $f \neq g$

Substitution $E \uplus \{x = t\} \Rightarrow_{\mathrm{SU}} E\{x \mapsto t\} \cup \{x = t\}$

if $x \in \mathit{vars}(E)$ and $x \notin \mathit{vars}(t)$

Occurs Check $E \uplus \{x = t\} \Rightarrow_{\mathrm{SU}} \bot$

if $x \neq t$ and $x \in \mathit{vars}(t)$

Orient $E \uplus \{t = x\} \Rightarrow_{\mathrm{SU}} E \cup \{x = t\}$

if $t \notin \mathcal{X}$

3.7.2 Theorem (Soundness, Completeness and Termination of $\Rightarrow_{\text{SU}}$)

If s, t are two terms with $\text{sort}(s) = \text{sort}(t)$ then

1. if $\{s = t\} \Rightarrow^*_{\text{SU}} E$ then any equation $(s' = t') \in E$ is well-sorted, i.e., $\text{sort}(s') = \text{sort}(t')$.
2. $\Rightarrow_{\text{SU}}$ terminates on $\{s = t\}$.
3. if $\{s = t\} \Rightarrow^*_{\text{SU}} E$ then σ is a unifier (mgu) of E iff σ is a unifier (mgu) of $\{s = t\}$.
4. if $\{s = t\} \Rightarrow^*_{\text{SU}} \perp$ then s and t are not unifiable.
5. if $\{s = t\} \Rightarrow^*_{\text{SU}} \{x_1 = t_1, \ldots, x_n = t_n\}$ and this is a normal form, then $\{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}$ is an mgu of s, t.

Size of Unification Problems

Any normal form of the unification problem E given by

$$\{f(x_1, g(x_1, x_1), x_3, \dots, g(x_n, x_n)) = f(g(x_0, x_0), x_2, g(x_2, x_2), \dots, x_{n+1})\}$$

with respect to $\Rightarrow_{\mathrm{SU}}$ is exponentially larger than E.

Polynomial Unification

The second calculus, polynomial unification, prevents the problem of exponential growth by introducing an implicit representation for the mgu.

For this calculus the size of a normal form is always polynomial in the size of the input unification problem.

Tautology $E \uplus \{t = t\} \;\Rightarrow_{\mathrm{PU}}\; E$

Decomposition $E \uplus \{f(s_1, \ldots, s_n) = f(t_1, \ldots, t_n)\} \;\Rightarrow_{\mathrm{PU}}$
$E \uplus \{s_1 = t_1, \ldots, s_n = t_n\}$

Clash $E \uplus \{f(t_1, \ldots, t_n) = g(s_1, \ldots, s_m)\} \;\Rightarrow_{\mathrm{PU}}\; \perp$
if $f \neq g$

Occurs Check $E \uplus \{x = t\} \Rightarrow_{\text{PU}} \bot$
if $x \neq t$ and $x \in vars(t)$

Orient $E \uplus \{t = x\} \Rightarrow_{\text{PU}} E \uplus \{x = t\}$
if $t \notin \mathcal{X}$

Substitution $E \uplus \{x = y\} \Rightarrow_{\text{PU}} E\{x \mapsto y\} \uplus \{x = y\}$
if $x \in vars(E)$ and $x \neq y$

Cycle $E \uplus \{x_1 = t_1, \ldots, x_n = t_n\} \;\Rightarrow_{\text{PU}}\; \bot$
if there are positions p_i with $t_i|_{p_i} = x_{i+1}, t_n|_{p_n} = x_1$ and some $p_i \neq \epsilon$

Merge $E \uplus \{x = t, x = s\} \;\Rightarrow_{\text{PU}}\; E \uplus \{x = t, t = s\}$
if $t, s \notin \mathcal{X}$ and $|t| \leq |s|$

3.7.4 Theorem (Soundness, Completeness and Termination of $\Rightarrow_{\text{PU}}$)

If s, t are two terms with $\text{sort}(s) = \text{sort}(t)$ then

1. if $\{s = t\} \Rightarrow^*_{\text{PU}} E$ then any equation $(s' = t') \in E$ is well-sorted, i.e., $\text{sort}(s') = \text{sort}(t')$.
2. $\Rightarrow_{\text{PU}}$ terminates on $\{s = t\}$.
3. if $\{s = t\} \Rightarrow^*_{\text{PU}} E$ then σ is a unifier (mgu) of E iff σ is a unifier (mgu) of $\{s = t\}$.
4. if $\{s = t\} \Rightarrow^*_{\text{PU}} \bot$ then s and t are not unifiable.

3.7.5 Theorem (Normal Forms Generated by $\Rightarrow_{\mathrm{PU}}$)

Let $\{s = t\} \Rightarrow^*_{\mathrm{PU}} \{x_1 = t_1, \ldots, x_n = t_n\}$ be a normal form. Then

1. $x_i \neq x_j$ for all $i \neq j$ and without loss of generality $x_i \notin \mathrm{vars}(t_{i+k})$ for all i, k, $1 \leq i < n$, $i + k \leq n$.
2. the substitution $\{x_1 \mapsto t_1\}\{x_2 \mapsto t_2\} \ldots \{x_n \mapsto t_n\}$ is an mgu of $s = t$.

First-Order Superposition

Now the result for ground superposition are lifted to superposition on first-order clauses with variables, still without equality.

The completeness proof of ground superposition above talks about (strictly) maximal literals of ground clauses. The non-ground calculus considers those literals that correspond to (strictly) maximal literals of ground instances.

The used ordering is exactly the ordering of Definition 3.12.1 where clauses with variables are projected to their ground instances for ordering computations.

3.13.1 Definition (Maximal Literal)

A literal L is called *maximal* in a clause C if and only if there exists a grounding substitution σ so that $L\sigma$ is maximal in $C\sigma$, i.e., there is no different $L' \in C$: $L\sigma \prec L'\sigma$. The literal L is called *strictly maximal* if there is no different $L' \in C$ such that $L\sigma \preceq L'\sigma$.

Note that the orderings KBO and LPO cannot be total on atoms with variables, because they are stable under substitutions. Therefore, maximality can also be defined on the basis of absence of greater literals. A literal L is called *maximal* in a clause C if $L \not\prec L'$ for all other literals $L' \in C$. It is called *strictly maximal* in a clause C if $L \not\preceq L'$ for all other literals $L' \in C$.

Superposition Left

$(N \uplus \{C_1 \vee P(t_1,\dots,t_n), C_2 \vee \neg P(s_1,\dots,s_n)\}) \;\Rightarrow_{\mathrm{SUP}}$
$(N \cup \{C_1 \vee P(t_1,\dots,t_n), C_2 \vee \neg P(s_1,\dots,s_n)\} \cup \{(C_1 \vee C_2)\sigma\})$
where (i) $P(t_1,\dots,t_n)\sigma$ is strictly maximal in $(C_1 \vee P(t_1,\dots,t_n))\sigma$ (ii) no literal in $C_1 \vee P(t_1,\dots,t_n)$ is selected (iii) $\neg P(s_1,\dots,s_n)\sigma$ is maximal and no literal selected in $(C_2 \vee \neg P(s_1,\dots,s_n))\sigma$, or $\neg P(s_1,\dots,s_n)$ is selected in $(C_2 \vee \neg P(s_1,\dots,s_n))\sigma$ (iv) σ is the mgu of $P(t_1,\dots,t_n)$ and $P(s_1,\dots,s_n)$

Factoring

$(N \uplus \{C \vee P(t_1,\dots,t_n) \vee P(s_1,\dots,s_n)\}) \;\Rightarrow_{\mathrm{SUP}}$
$(N \cup \{C \vee P(t_1,\dots,t_n) \vee P(s_1,\dots,s_n)\} \cup \{(C \vee P(t_1,\dots,t_n))\sigma\})$
where (i) $P(t_1,\dots,t_n)\sigma$ is maximal in $(C \vee P(t_1,\dots,t_n) \vee P(s_1,\dots,s_n))\sigma$ (ii) no literal is selected in $C \vee P(t_1,\dots,t_n) \vee P(s_1,\dots,s_n)$ (iii) σ is the mgu of $P(t_1,\dots,t_n)$ and $P(s_1,\dots,s_n)$

Note that the above inference rules Superposition Left and Factoring are generalizations of their respective counterparts from the ground superposition calculus above. Therefore, on ground clauses they coincide. Therefore, we can safely overload them in the sequel.

3.13.3 Definition (Abstract Redundancy)

A clause C is *redundant* with respect to a clause set N if for all ground instances $C\sigma$ there are clauses $\{C_1, \ldots, C_n\} \subseteq N$ with ground instances $C_1\tau_1, \ldots, C_n\tau_n$ such that $C_i\tau_i \prec C\sigma$ for all i and $C_1\tau_1, \ldots, C_n\tau_n \models C\sigma$.

3.13.4 Definition (Saturation)

A set N of clauses is called *saturated up to redundancy*, if any inference from non-redundant clauses in N yields a redundant clause with respect to N or is contained in N.

In contrast to the ground case, the above abstract notion of redundancy is not effective, i.e., it is undecidable for some clause C whether it is redundant, in general. Nevertheless, the concrete ground redundancy notions carry over to the non-ground case. Note also that a clause C is contained in N modulo renaming of variables.

Subsumption $(N \uplus \{C_1, C_2\}) \Rightarrow_{\text{SUP}} (N \cup \{C_1\})$
provided $C_1\sigma \subset C_2$ for some σ

Tautology Deletion $(N \uplus \{C \vee P(t_1, \ldots, t_n) \vee \neg P(t_1, \ldots, t_n)\})$
$\Rightarrow_{\text{SUP}} (N)$

Let rdup be a function from clauses to clauses that removes duplicate literals, i.e., $\mathrm{rdup}(C) = C'$ where $C' \subseteq C$, C' does not contain any duplicate literals, and for each $L \in C$ also $L \in C'$.

Condensation $(N \uplus \{C_1 \lor L \lor L'\}) \Rightarrow_{\mathrm{SUP}}$
$(N \cup \{\mathrm{rdup}((C_1 \lor L \lor L')\sigma)\})$
provided $L\sigma = L'$ and $\mathrm{rdup}((C_1 \lor L \lor L')\sigma)$ subsumes $C_1 \lor L \lor L'$ for some σ

Subsumption Resolution $(N \uplus \{C_1 \lor L, C_2 \lor L'\}) \Rightarrow_{\mathrm{SUP}}$
$(N \cup \{C_1 \lor L, C_2\})$
where $L\sigma = \neg L'$ and $C_1\sigma \subseteq C_2$ for some σ

3.13.7 Lemma (Lifting)

Let $D \vee L$ and $C \vee L'$ be variable-disjoint clauses and σ a grounding substitution for $C \vee L$ and $D \vee L'$. If there is a superposition left inference
$(N \uplus \{(D \vee L)\sigma, (C \vee L')\sigma\}) \Rightarrow_{\text{SUP}}$
$(N \cup \{(D \vee L)\sigma, (C \vee L')\sigma\} \cup \{D\sigma \vee C\sigma\})$ and if
$\text{sel}((D \vee L)\sigma) = \text{sel}((D \vee L))\sigma$, $\text{sel}((C \vee L')\sigma) = \text{sel}((C \vee L'))\sigma$, then there exists a mgu τ such that
$(N \uplus \{D \vee L, C \vee L'\}) \Rightarrow_{\text{SUP}} (N \cup \{D \vee L, C \vee L'\} \cup \{(D \vee C)\tau\})$.

Let $C \vee L \vee L'$ be a clause and σ a grounding substitution for $C \vee L \vee L'$. If there is a factoring inference
$(N \uplus \{(C \vee L \vee L')\sigma\}) \Rightarrow_{\text{SUP}} (N \cup \{(C \vee L \vee L')\sigma\} \cup \{(C \vee L)\sigma\})$
and if $\text{sel}((C \vee L \vee L')\sigma) = \text{sel}((C \vee L \vee L'))\sigma$, then there exists a mgu τ such that
$(N \uplus \{C \vee L \vee L'\}) \Rightarrow_{\text{SUP}} (N \cup \{C \vee L \vee L'\} \cup \{(C \vee L)\tau\})$

3.13.8 Example (First-Order Reductions are not Liftable)

Consider the two clauses $P(x) \lor Q(x)$, $P(g(y))$ and grounding substitution $\{x \mapsto g(a), y \mapsto a\}$. Then $P(g(y))\sigma$ subsumes $(P(x) \lor Q(x))\sigma$ but $P(g(y))$ does not subsume $P(x) \lor Q(x)$. For all other reduction rules similar examples can be constructed.

3.13.9 Lemma (Soundness and Completeness)

First-Order Superposition is sound and complete.

3.13.10 Lemma (Redundant Clauses are Obsolete)

If a clause set N is unsatisfiable, then there is a derivation $N \Rightarrow^*_{\text{SUP}} N'$ such that $\bot \in N'$ and no clause in the derivation of $\bot$ is redundant.

3.13.11 Lemma (Model Property)

If N is a saturated clause set and $\bot \notin N$ then $\operatorname{grd}(\Sigma, N)_{\mathcal{I}} \models N$.

Algorithm 1: SupProver(N)

```
Input  : A set of clauses $N$.
Output : A saturated set of clauses $N'$, equivalent to $N$.
WO := $\emptyset$;
US := $N$;
while (US $\neq \emptyset$ and $\bot \notin$ US) do
    Given:= pick a clause from US;
    WO := WO $\cup$ {Given};
    New := SupLeft(WO,Given) $\cup$ Fact(Given);
    while (New $\neq \emptyset$) do
        Given:= pick a clause from New;
        if (!TautDel(Given)) then
            if (!SubDel(Given,WO $\cup$US)) then
                Given:= Cond(Given);
                Given:= SubRes(Given,WO);
                WO:= SubDel(WO,Given);
                US:= SubDel(US,Given);
                New:= New $\cup$ SubRes(WO $\cup$US,Given);
                US:= US $\cup$ {Given };
    end
end
return WO;
```