Automated Reasoning

Lecture 11: Unification

Jacques Fleuriot
`jdf@inf.ed.ac.uk`

Recap

- This lecture:
  - Solving equations by Unification
  - Matching and Unification algorithms
  - Building-in axioms: E-Unification

Motivation

Unification: **finding a common instance of two terms**

Informally: we want to make two terms **identical** by finding the **most general substitution** of terms for variables.

Why?

- Applying rules in Isabelle: working out what $?P, ?Q, ?x$ are
- Heavily used in automated first-order theorem proving to postpone decisions during proof search: PROLOG, tableau provers, resolution provers
- Also used in most type inference algorithms (Haskell, OCaml, SML, Scala, ...)

A First Look at Unification

Unification: **finding a common instance of two terms**

Informally: we want to make two terms **identical** by finding the **most general substitution** of terms for variables.

Example

Can we make these pairs of terms equal by finding a common instance (assuming X, Y are variables and a, b are constants)?

$f(X, b)$ and $f(a, Y)$	Yes: $[a/X, b/Y]$	instance: $f(a, b)$
$f(X, X)$ and $f(a, b)$	No	
$f(X, X)$ and $f(Y, g(Y))$	No	

Only (meta-)variables (X, Y, Z, ...) can be replaced by other terms.

Matching

Problem

*Given **pattern** and **target** find a **substitution** such that:*

$$\mathit{pattern}[\mathit{substitution}] \equiv \mathit{target}$$

where $\equiv$ *means that the terms are identical.*

Example

$$(s(X) + Y)[0/X, s(0)/Y] \equiv (s(0) + s(0))$$

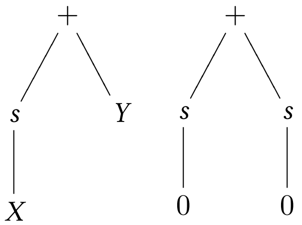
How we do find an adequate substitution?

We view matching as equation solving.

Matching (continued)

Discover a substition by decomposing the equation to be solved along the term trees:

$$
\begin{gathered}
(s(X) + Y) \equiv (s(0) + s(0)) \\
\downarrow \\
(s(X) \equiv s(0)) \wedge (Y \equiv s(0)) \\
\downarrow \\
(X \equiv 0) \wedge (Y \equiv s(0))
\end{gathered}
$$



Some Abbreviations

Term	Meaning
$\overrightarrow{t}$	$t_1, \ldots, t_n \quad (t \geq 1)$
$\bigwedge_i t_i$	$t_1 \wedge \ldots \wedge t_n$
$\mathrm{vars}(t)$	the set of free variables in t
Vars	the set of (all) free variables

$$\mathrm{vars}(f(X, Y, g(a, Z, X))) = \{X, Y, Z\}$$

$$\mathrm{vars}(f(a, b, c)) = \{\}$$

Matching as Equation Solving

Start with the *pattern* and *target* **standardised apart**:

$$\mathrm{vars}(\mathit{pattern}) \cap \mathrm{vars}(\mathit{target}) = \{\}$$

Goal is to solve for *vars*(*pattern*) in equation $\mathit{pattern} \equiv \mathit{target}$.

Strategy is to use transformation rules:

$$\begin{array}{c} \mathit{pattern} \equiv \mathit{target} \\ \downarrow \\ \vdots \\ \downarrow \\ X_1 \equiv t_1 \wedge \ldots \wedge X_n \equiv t_n \end{array}$$

Resulting substitution is $[t_1/X_1, \ldots, t_n/X_n]$.

Transformations end in failure if no match is possible.

Transformation Rules for Matching (Examples)

Decompose	$s(X) + Y \equiv s(0) + s(0)$ $\downarrow$ $s(X) \equiv s(0) \wedge Y \equiv s(0)$	
Conflict	$s(X) + y \equiv s(0)$ $\downarrow$ fail	Cannot match: $s \not\equiv +$
Eliminate	$(X + Y \equiv s(0) + 0) \wedge (Y \equiv 0)$ $\downarrow$ $(X + 0 \equiv s(0) + 0) \wedge (Y \equiv 0)$	
Delete	$X \equiv 0 \wedge (s(0) + 0 \equiv s(0) + 0)$ $\downarrow$ $X \equiv 0$	

Transformation Rules for Matching

Assumptions: s and t are arbitrary terms and are standardised apart.

Name	**Before**	**After**	**Condition**
Decompose	$P \wedge f(\overrightarrow{s}) \equiv f\left(\overrightarrow{t}\right)$	$P \wedge \bigwedge_i s_i \equiv t_i$	
Conflict	$P \wedge f(\overrightarrow{s}) \equiv g\left(\overrightarrow{t}\right)$	fail	$f \neq g$
Eliminate	$P \wedge X \equiv t$	$P[t/X] \wedge X \equiv t$	$X \in \text{vars}(P)$
Delete	$P \wedge t \equiv t$	P	

Algorithm terminates when no further rules apply and fail has not occurred.

The algorithm terminates with a match iff there is one.

The algorithm may terminate without a match: e.g., $X \equiv a \wedge b \equiv Y$

Unification

Unification is two-way matching (there is no distinction between pattern and target).

$$term_1[substitution] \equiv term_2[substitution]$$

Example

What substitution makes $(s(X) + s(0))$ and $(s(0) + Y)$ identical?

$$\theta = [0/X, s(0)/Y]$$

We need to add **extra** rules to the matching algorithm:

$$\begin{array}{cl}
(s(X) + s(0)) \equiv (s(0) + Y) & \\
\downarrow & \text{Decompose} \\
s(X) \equiv s(0) \wedge s(0) \equiv Y & \\
\downarrow & \text{Decompose} \\
X \equiv 0 \wedge s(0) \equiv Y & \\
\downarrow & \text{Switch} \\
X \equiv 0 \wedge Y \equiv s(0) &
\end{array}$$

New Transformation Rules

Switch

$$t \equiv X$$
$$\downarrow$$
$$X \equiv t$$

Switch rule applies only if *lhs* is not originally a variable

Coalesce

$$X \equiv Y + 1 \wedge Y \equiv X$$
$$\downarrow$$
$$X \equiv X + 1 \wedge Y \equiv X$$

Similar to Eliminate, except both *lhs* and *rhs* are variables

Occurs Check

$$X \equiv X + 1$$
$$\downarrow$$
$$\text{fail}$$

lhs cannot occur in *rhs*

Example

$$\begin{array}{c} f(X, X) \equiv f(Y, Y+1) \\ \downarrow \text{Decompose} \\ X \equiv Y \wedge X \equiv Y + 1 \\ \downarrow \text{Coalesce} \\ X \equiv Y \wedge Y \equiv Y + 1 \\ \downarrow \text{Occurs check} \\ \text{fail} \end{array}$$

$$\begin{array}{c} p(X) \wedge X \equiv X + 1 \\ \downarrow \text{Eliminate} \\ p(X+1) \wedge X \equiv X + 1 \\ \downarrow \text{Eliminate} \\ p((X+1)+1) \wedge X \equiv X + 1 \\ \downarrow \text{Eliminate} \\ \ldots \end{array}$$

Non-termination can result without the occurs check.

Unification Algorithm

Assumptions: s and t are arbitrary terms and $Vars = vars(s) \cup vars(t)$.

Name	Before	After	Condition
Decompose	$P \wedge f(\overrightarrow{s}) \equiv f\left(\overrightarrow{t}\right)$	$P \wedge \bigwedge_i s_i \equiv t_i$	
Conflict	$P \wedge f(\overrightarrow{s}) \equiv g\left(\overrightarrow{t}\right)$	fail	$f \not\equiv g$
Switch	$P \wedge s \equiv X$	$P \wedge X \equiv s$	$X \in Vars$ $s \notin Vars$
Delete	$P \wedge s \equiv s$	P	
Eliminate	$P \wedge X \equiv s$	$P[s/X] \wedge X \equiv s$	$X \in vars(P)$ $X \notin vars(s)$ $s \notin Vars$
Occurs Check	$P \wedge X \equiv s$	fail	$X \in vars(s)$ $s \notin Vars$
Coalesce	$P \wedge X \equiv Y$	$P[Y/X] \wedge X \equiv Y$	$X, Y \in vars(P)$ $X \not\equiv Y$

- Conditions ensure that at most one rule applies to each conjunct
- Algorithm terminates with success when no further rules apply.

Composition of Unifiers (Substitutions)

Definition

If ϕ and θ are substitutions then their *composition* $\phi \circ \theta$ is also a substitution which, for any term t, satisfies the following property:

$$t[\phi \circ \theta] \equiv (t[\phi])[\theta]$$

Composition of Unifiers (Substitutions)

Definition

If ϕ and θ are substitutions then their *composition* $\phi \circ \theta$ is also a substitution which, for any term t, satisfies the following property:

$$t[\phi \circ \theta] \equiv (t[\phi])[\theta]$$

Examples:

$$[a/x] \circ [b/y] = [a/x, b/y]$$

Composition of Unifiers (Substitutions)

Definition

If ϕ and θ are substitutions then their *composition* $\phi \circ \theta$ is also a substitution which, for any term t, satisfies the following property:

$$t[\phi \circ \theta] \equiv (t[\phi])[\theta]$$

Examples:

$$[a/x] \circ [b/y] = [a/x, b/y]$$
$$[\mathrm{g}(y)/x] \circ [b/y] = [\mathrm{g}(b)/x, b/y]$$

Composition of Unifiers (Substitutions)

Definition

If ϕ and θ are substitutions then their *composition* $\phi \circ \theta$ is also a substitution which, for any term t, satisfies the following property:

$$t[\phi \circ \theta] \equiv (t[\phi])[\theta]$$

Examples:

$$[a/x] \circ [b/y] = [a/x, b/y]$$
$$[\mathrm{g}(y)/x] \circ [b/y] = [\mathrm{g}(b)/x, b/y]$$
$$[a/x] \circ [b/x] = [a/x]$$

Composition of Unifiers (Substitutions)

Definition

If ϕ and θ are substitutions then their *composition* $\phi \circ \theta$ is also a substitution which, for any term t, satisfies the following property:

$$t[\phi \circ \theta] \equiv (t[\phi])[\theta]$$

Examples:

$$\begin{aligned}
&[a/x] \circ [b/y] = [a/x, b/y] \\
&[g(y)/x] \circ [b/y] = [g(b)/x, b/y] \\
&[a/x] \circ [b/x] = [a/x]
\end{aligned}$$

- Equality of substitutions: $\phi = \theta$ if $x[\phi] = x[\theta]$ for any variable x.
- Properties: $(\phi \circ \theta) \circ \sigma = \phi \circ (\theta \circ \sigma)$, $\phi \circ [] = \phi$ and $[] \circ \phi = \phi$.
- Composition is needed to define the notion of a *most general unifier.*

Properties of the Unification Algorithm

- The algorithm will find a unifier, if it exists.
- It returns the **most general unifier** (mgu) θ.

  Definition

  Given any two terms s and t, θ is their mgu if:

  $$s[\theta] \equiv t[\theta] \wedge \forall \phi.\ s[\phi] \equiv t[\phi] \rightarrow \exists \psi.\ \phi = \theta \circ \psi.$$

  Consider $g(g(X))$ and $g(Y)$. Is $[g(3)/Y, 3/X]$ a unifier? Is it the mgu?
- mgu is **unique** up to alphabetic variance;
- the algorithm can easily be extended to simultaneous unification on n expressions.

Building-in Axioms

General Scheme:

$$(Ax_1 \cup Ax_2) + unif \Longrightarrow Ax_1 + unif_{Ax_2}.$$

Some axioms of the theory become built into unification.

Example

Commutative-Unification

$$\begin{array}{c} X + 2 = Y + 3 \\ \downarrow \\ Y = 2 \wedge X = 3 \end{array} \qquad \text{We no longer use } \equiv \text{ but } =$$

How do we deal with this?
We can add a new transformation rule (**Mutate rule**).

Unification Algorithm for Commutativity

Name	Before	After	Condition
Decompose	$P \wedge f(\overrightarrow{s}) = f(\overrightarrow{t})$	$P \wedge \bigwedge_i s_i = t_i$	
Conflict	$P \wedge f(\overrightarrow{s}) = g(\overrightarrow{t})$	fail	$f \neq g$
Switch	$P \wedge s = X$	$P \wedge X = s$	$X \in Vars$ $s \notin Vars$
Delete	$P \wedge s = s$	P	
Eliminate	$P \wedge X = s$	$P[s/X] \wedge X = s$	$X \in vars(P)$ $X \notin vars(s)$ $s \notin Vars$
Check	$P \wedge X = s$	fail	$X \in vars(s)$ $s \notin Vars$
Coalesce	$P \wedge X = Y$	$P[Y/X] \wedge X = Y$	$X, Y \in vars(P)$ $X \neq Y$
Mutate	$P \wedge f(s_1, t_1) = f(s_2, t_2)$	$P \wedge s_1 = t_2 \wedge t_1 = s_2$	f is commutative

Decompose and Mutate rules overlap.

Most General Unifiers

For ordinary unification, the mgu is unique, but what happens when new rules are built-into the unification algorithm?

Multiple mgus: Commutative unification

$$X + Y = a + b \longrightarrow \begin{cases} X = a \wedge Y = b \\ X = b \wedge Y = a \end{cases}$$

Both are equally general.

Infinitely many mgus: Associative unification $X + (Y + Z) = (X + Y) + Z$.

$$X + a = a + X \longrightarrow \begin{cases} X = a \\ X = a + a \\ X = a + a + a \\ \ldots \end{cases}$$

All independent (not unifiable).

No mgus: Build in $f(0, X) = X$ and $g(f(X, Y)) = g(Y)$:

$$g(X) = g(a) \longrightarrow \begin{cases} X = a \\ X = f(Y_1, a) \\ X = f(Y_1, f(Y_2, a)) \end{cases}$$

Many unifiers but no mgu.

Types of Unification

Unitary	A single unique mgu, or none (predicate logic).
Finitary	Finite number of mgus (predicate logic with commutativity).
Infinitary	Possibly infinite number of mgus (predicate logic with associativity).
Nullary	No mgus exist, although unifiers may exist.
Undecidable	Unification not decidable — no algorithm.

Types of Unification

Axioms	Type	Decidable
nil	unitary	yes
commutative	finitary	yes
associative	infinitary	yes
assoc. + dist.	infinitary	yes
lambda calculus	infinitary	no
λ-calculus pattern fragment	unitary	yes

Summary

- Unification (Bundy Ch. 17.1 - 17.4)
  - Algorithms for matching and unification.
  - Unification as equation solving.
  - Transformation rules for equation solving.
  - Building-in axioms.(E-Unification/Semantic Unification)
  - Most general unifiers and classification.
- Next time: Proof by rewriting