Programming Languages and Compilers (CS 421)

Elsa L Gunter
2112 SC, UIUC

http://courses.engr.illinois.edu/cs421

Based in part on slides by Mattox Beckman, as updated by Vikram Adve and Gul Agha
Background for Unification

- **Terms** made from **constructors** and **variables** (for the simple first order case)
- Constructors may be applied to arguments (other terms) to make new terms
- Variables and constructors with no arguments are base cases
- Constructors applied to different number of arguments (arity) considered different
- **Substitution** of terms for variables
type term = Variable of string
    | Const of (string * term list)

let rec subst var_name residue term =
    match term with
    Variable name ->
      if var_name = name then residue else term
    | Const (c, tys) ->
      Const (c, List.map (subst var_name residue) tys)
Unification Problem

Given a set of pairs of terms ("equations")

\{(s_1, t_1), (s_2, t_2), \ldots, (s_n, t_n)\}

(the unification problem) does there exist
a substitution \(\sigma\) (the unification solution)
of terms for variables such that

\[\sigma(s_i) = \sigma(t_i),\]

for all \(i = 1, \ldots, n\)?
Uses for Unification

- Type Inference and type checking
- Pattern matching as in OCAML
  - Can use a simplified version of algorithm
- Logic Programming - Prolog
- Simple parsing
Unification Algorithm

- Let $S = \{(s_1 = t_1), (s_2 = t_2), \ldots, (s_n = t_n)\}$ be a unification problem.

- Case $S = \{\}$: $\text{Unif}(S) = \text{Identity function}$ (i.e., no substitution)

- Case $S = \{(s, t)\} \cup S'$: Four main steps
Unification Algorithm

- **Delete**: if \( s = t \) (they are the same term) then \( \text{Unif}(S) = \text{Unif}(S') \)

- **Decompose**: if \( s = f(q_1, \ldots, q_m) \) and \( t = f(r_1, \ldots, r_m) \) (same \( f \), same \( m ! \)), then \( \text{Unif}(S) = \text{Unif} \left( \{(q_1, r_1), \ldots, (q_m, r_m)\} \cup S' \right) \)

- **Orient**: if \( t = x \) is a variable, and \( s \) is not a variable, \( \text{Unif}(S) = \text{Unif} \left( \{(x = s)\} \cup S' \right) \)
Unification Algorithm

**Eliminate:** if \( s = x \) is a variable, and \( x \) does not occur in \( t \) (the occurs check), then

- Let \( \varphi = \{ x \rightarrow t \} \)
- Let \( \psi = \text{Unif}(\varphi(S')) \)
- \( \text{Unif}(S) = \{ x \rightarrow \psi(t) \} \circ \psi \)

**Note:** \( \{ x \rightarrow a \} \circ \{ y \rightarrow b \} = \{ y \rightarrow ((\{ x \rightarrow a \})(b)) \} \circ \{ x \rightarrow a \} \) if \( y \) not in \( a \)
Tricks for Efficient Unification

- Don’t return substitution, rather do it incrementally
- Make substitution be constant time
  - Requires implementation of terms to use mutable structures (or possibly lazy structures)
  - We won’t discuss these
Example

- $x, y, z$ variables, $f, g$ constructors

- Unify \{(f(x) = f(g(f(z), y))), (g(y, y) = x)\} = ?
Example

- \( x, y, z \) variables, \( f, g \) constructors
- \( S = \{(f(x) = f(g(f(z),y))), (g(y,y) = x)\} \) is nonempty

- Unify \( \{(f(x) = f(g(f(z),y))), (g(y,y) = x)\} = \? \)
Example

- \( x, y, z \) variables, \( f, g \) constructors
- Pick a pair: \( (g(y, y) = x) \)

- Unify \( \{(f(x) = f(g(f(z), y))), (g(y, y) = x)\} = ? \)
Example

- $x, y, z$ variables, $f, g$ constructors
- Pick a pair: $(g(y, y)) = x$
- Orient: $(x = g(y, y))$

- Unify \{(f(x) = f(g(f(z), y))), (g(y, y) = x)\} = \{f(x) = f(g(f(z), y)), (x = g(y, y))\}

by Orient
Example

- $x, y, z$ variables, $f, g$ constructors

- Unify \{(f(x) = f(g(f(z), y))), (x = g(y, y))\} = ?
Example

- $x, y, z$ variables, $f, g$ constructors
- $\{(f(x) = f(g(f(z), y))), (x = g(y, y))\}$ is non-empty

- Unify $\{(f(x) = f(g(f(z), y))), (x = g(y, y))\} = ?$
Example

- \(x, y, z\) variables, \(f, g\) constructors
- Pick a pair: \((x = g(y, y))\)

- Unify \{\((f(x) = f(g(f(z), y))), (x = g(y, y))\)\} = ?
Example

- $x, y, z$ variables, $f, g$ constructors
- Pick a pair: $(x = g(y, y))$
- Eliminate $x$ with substitution $\{x \rightarrow g(y, y)\}$
  - Check: $x$ not in $g(y, y)$
- Unify $\{(f(x) = f(g(f(z), y))), (x = g(y, y))\}$ = ?
Example

- \( x, y, z \) variables, \( f, g \) constructors
- Pick a pair: \((x = g(y, y))\)
- Eliminate \( x \) with substitution \( \{x \rightarrow g(y, y)\} \)

- Unify \( \{(f(x) = f(g(f(z), y))), (x = g(y, y))\} = \)
  Unify \( \{(f(g(y, y)) = f(g(f(z), y)))\} \)
  o \( \{x \rightarrow g(y, y)\} \)
Example

- $x, y, z$ variables, $f, g$ constructors

Unify \{(f(g(y,y)) = f(g(f(z),y)))\}

$\{x \mapsto g(y,y)\} = ?$
Example

- $x,y,z$ variables, $f,g$ constructors
- $\{(f(g(y,y)) = f(g(f(z),y)))\}$ is non-empty

- Unify $\{(f(g(y,y)) = f(g(f(z),y)))\}$
  $\circ \{x \mapsto g(y,y)\} = ?$
Example

- $x,y,z$ variables, $f,g$ constructors
- Pick a pair: $(f(g(y,y)) = f(g(f(z),y)))$

- Unify $\{(f(g(y,y)) = f(g(f(z),y)))\}$
  - $\{x \rightarrow g(y,y)\} = ?$
Example

- $x,y,z$ variables, $f,g$ constructors
- Pick a pair: $(f(g(y,y)) = f(g(f(z),y)))$
- Decompose: $(f(g(y,y)) = f(g(f(z),y)))$
  becomes $\{(g(y,y) = g(f(z),y))\}$

- Unify $\{(f(g(y,y)) = f(g(f(z),y)))\}$
  $\circ \{x \rightarrow g(y,y)\} =$
- Unify $\{(g(y,y) = g(f(z),y))\} \circ \{x \rightarrow g(y,y)\}$
Example

- $x,y,z$ variables, $f,g$ constructors
- $\{(g(y,y) = g(f(z),y))\}$ is non-empty

- Unify $\{(g(y,y) = g(f(z),y))\}$
  $\circ \{x \rightarrow g(y,y)\} = ?$
Example

- \(x, y, z\) variables, \(f, g\) constructors
- Pick a pair: \((g(y, y) = g(f(z), y))\)

- Unify \((g(y, y) = g(f(z), y))\)
  \(\{x \rightarrow g(y, y)\} = ?\)
Example

- \(x, y, z\) variables, \(f, g\) constructors
- Pick a pair: \((f(g(y,y)) = f(g(f(z),y)))\)
- Decompose: \((g(y,y)) = g(f(z),y))\) becomes \({(y = f(z)); (y = y)}\)

- Unify \({(g(y,y) = g(f(z),y))}\) o \({x \rightarrow g(y,y)}\) =
  Unify \({(y = f(z)); (y = y)}\) o \({x \rightarrow g(y,y)}\)
Example

- $x, y, z$ variables, $f, g$ constructors

- Unify $\{(y = f(z)); (y = y)\} \circ \{x \rightarrow g(y, y)\} = ?$
Example

- $x, y, z$ variables, $f, g$ constructors
- $\{(y = f(z)); (y = y)\} \circ \{x \mapsto g(y,y)\}$ is non-empty
- Unify $\{(y = f(z)); (y = y)\} \circ \{x \mapsto g(y,y)\} = ?$
Example

- $x, y, z$ variables, $f, g$ constructors
- Pick a pair: $(y = f(z))$

- Unify $\{(y = f(z)); (y = y)\} \circ \{x \rightarrow g(y, y)\} = ?$
Example

- $x, y, z$ variables, $f, g$ constructors
- Pick a pair: $(y = f(z))$
- Eliminate $y$ with $\{y \rightarrow f(z)\}$
- Unify $\{(y = f(z)); (y = y)\} \circ \{x \rightarrow g(y, y)\} = \{(f(z) = f(z))\}$
  - $\circ \{y \rightarrow f(z)\} \circ \{x \rightarrow g(y, y)\} = \{(f(z) = f(z))\}$
  - $\circ \{y \rightarrow f(z); x \rightarrow g(f(z), f(z))\} = \{(f(z) = f(z))\}$
Example

- \(x, y, z\) variables, \(f, g\) constructors

Unify \(\{(f(z) = f(z))\}\)

\(\circ \{y \rightarrow f(z); x \rightarrow g(f(z), f(z))\} = ?\)
Example

- $x, y, z$ variables, $f, g$ constructors
- $\{(f(z) = f(z))\}$ is non-empty

Unify $\{(f(z) = f(z))\}$
  $\circ \{y \rightarrow f(z); x \rightarrow g(f(z), f(z))\} = ?$
Example

- $x, y, z$ variables, $f, g$ constructors
- Pick a pair: $(f(z) = f(z))$

Unify $\{(f(z) = f(z))\}$

$\{y \rightarrow f(z); x \rightarrow g(f(z), f(z))\} = ?$
Example

- $x, y, z$ variables, $f, g$ constructors
- Pick a pair: $(f(z) = f(z))$
- Delete
- Unify $\{(f(z) = f(z))\}$
  - $\circ \{y \rightarrow f(z); x \rightarrow g(f(z), f(z))\} =$
  - Unify $\{} \circ \{y \rightarrow f(z); x \rightarrow g(f(z), f(z))\}$
Example

- $x, y, z$ variables, $f, g$ constructors

Unify $\emptyset \circ \{ y \rightarrow f(z); x \rightarrow g(f(z), f(z)) \} = ?$
Example

- \(x, y, z\) variables, \(f, g\) constructors
- \(\{\}\) is empty
- Unify \(\{\}\) = identity function
- Unify \(\{\} \circ \{y \rightarrow f(z); x \rightarrow g(f(z), f(z))\}\) = \(\{y \rightarrow f(z); x \rightarrow g(f(z), f(z))\}\)
Example

- Unify \{ (f(x) = f(g(f(z),y))), (g(y,y) = x) \} = \{ y \rightarrow f(z); x \rightarrow g(f(z), f(z)) \}

\[
\begin{align*}
f(x) &= f(g(f(z), y)) \\
\rightarrow f(g(f(z), f(z))) &= f(g(f(z), f(z)))
\end{align*}
\]

\[
\begin{align*}
g(y, y) &= x \\
\rightarrow g(f(z), f(z)) &= g(f(z), f(z))
\end{align*}
\]
Example of Failure: Decompose

- Unify\{((f(x,g(y)) = f(h(y),x))}\}
- Decompose: \((f(x,g(y)) = f(h(y),x))\)
- = Unify \{(x = h(y)), (g(y) = x)\}
- Orient: \((g(y) = x)\)
- = Unify \{(x = h(y)), (x = g(y))\}
- Eliminate: \((x = h(y))\)
- Unify \{((h(y) = g(y))) \circ \{x \rightarrow h(y)\}\}
- No rule to apply! Decompose fails!
Example of Failure: Occurs Check

- Unify\{(f(x,g(x)) = f(h(x),x))\}
- Decompose: \( (f(x,g(x)) = f(h(x),x)) \)
- \( = \) Unify \{(x = h(x)), (g(x) = x)\}
- Orient: \( (g(y) = x) \)
- \( = \) Unify \{(x = h(x)), (x = g(x))\}
- No rules apply.
Major Phases of a Compiler

Source Program
Lex
Tokens
Parse
Abstract Syntax
Semantic Analysis
Symbol Table
Translate
Intermediate Representation

Optimize
Optimized IR
Instruction Selection
Unoptimized Machine-Specific Assembly Language
Optimize
Optimized Machine-Specific Assembly Language
Emit code
Assembly Language
Assembler

Relocatable Object Code
Linker
Machine Code

Modified from “Modern Compiler Implementation in ML”, by Andrew Appel
Meta-discourse

- Language Syntax and Semantics
- Syntax
  - Regular Expressions, DFSAs and NDFSAs
  - Grammars
- Semantics
  - Natural Semantics
  - Transition Semantics
Language Syntax

- Syntax is the description of which strings of symbols are meaningful expressions in a language
- It takes more than syntax to understand a language; need meaning (semantics) too
- Syntax is the entry point
Syntax of English Language

- **Pattern 1**

<table>
<thead>
<tr>
<th>Subject</th>
<th>Verb</th>
</tr>
</thead>
<tbody>
<tr>
<td>David</td>
<td>sings</td>
</tr>
<tr>
<td>The dog</td>
<td>barked</td>
</tr>
<tr>
<td>Susan</td>
<td>yawned</td>
</tr>
</tbody>
</table>

- **Pattern 2**

<table>
<thead>
<tr>
<th>Subject</th>
<th>Verb</th>
<th>Direct Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>David</td>
<td>sings</td>
<td>ballads</td>
</tr>
<tr>
<td>The professor</td>
<td>wants</td>
<td>to retire</td>
</tr>
<tr>
<td>The jury</td>
<td>found</td>
<td>the defendant guilty</td>
</tr>
</tbody>
</table>
Elements of Syntax

- Character set – previously always ASCII, now often 64 character sets
- Keywords – usually reserved
- Special constants – cannot be assigned to
- Identifiers – can be assigned to
- Operator symbols
- Delimiters (parenthesis, braces, brackets)
- Blanks (aka white space)
Elements of Syntax

- Expressions
  
  if ... then begin ... ; ... end else begin ... ; ... end

- Type expressions
  
  \[ \text{typexpr}_1 \rightarrow \text{typexpr}_2 \]

- Declarations (in functional languages)
  
  let pattern_1 = expr_1 in expr

- Statements (in imperative languages)
  
  a = b + c

- Subprograms
  
  let pattern_1 = let rec inner = ... in expr
Elements of Syntax

- Modules
- Interfaces
- Classes (for object-oriented languages)
Lexing and Parsing

- Converting strings to abstract syntax trees done in two phases
  - **Lexing**: Converting string (or streams of characters) into lists (or streams) of tokens (the “words” of the language)
    - Specification Technique: Regular Expressions
  - **Parsing**: Convert a list of tokens into an abstract syntax tree
    - Specification Technique: BNF Grammars
Formal Language Descriptions

- Regular expressions, regular grammars, finite state automata

- Context-free grammars, BNF grammars, syntax diagrams

- Whole family more of grammars and automata – covered in automata theory
Grammars

- Grammars are formal descriptions of which strings over a given character set are in a particular language
- Language designers write grammar
- Language implementers use grammar to know what programs to accept
- Language users use grammar to know how to write legitimate programs