CS375: Logic and Theory of Computing

Fuhua (Frank) Cheng

Department of Computer Science
University of Kentucky
Table of Contents:

- Week 1: Preliminaries (set algebra, relations, functions) (read Chapters 1-4)
- Weeks 3-6: Regular Languages, Finite Automata (Chapter 11)
- Weeks 7-9: Context-Free Languages, Pushdown Automata (Chapters 12)
- Weeks 10-12: Turing Machines (Chapter 13)
Table of Contents (conti):

- Weeks 13-14: Propositional Logic (Chapter 6), Predicate Logic (Chapter 7), Computational Logic (Chapter 9), Algebraic Structures (Chapter 10)
Robinson’s Unification Algorithm:
For a finite set $S$ of atoms find whether $S$ has an mgu.

1. $k := 0; \ \theta_0 := \varepsilon; \ \text{go to Step 2.}$
2. If $S\theta_k$ is a singleton then stop with mgu $\theta_k$
   Otherwise construct $D_k$ (set of terms in leftmost position of disagreement); go to Step 3.
3. If $D_k$ has a variable $v$ and a term $t$ such that $v$ does not occur in $t$ then
   \[ \theta_{k+1} := \theta_k[v/t]; \quad k := k + 1; \quad \text{go to Step 2.} \]
   Otherwise stop ($S$ is not unifiable).
Disagreement set (DS):

If $S$ is a set of literals, then the \textit{disagreement set} of
$S$ is constructed in the following way:

1. Find the \textbf{longest common substring} that starts at
the left end of each literal of $S$

Example: $S = \{p(x, f(x), y), p(x, y, z), p(x, f(a), b)\}$

Longest common substring: $p(x,$
(of length 4)
Disagreement set (DS):

2. The disagreement set of $S$ is the set of all the terms that occur in the literals of $S$ that are immediately to the right of the longest common substring.

Example: $S = \{ p(x, f(x), y), p(x, y, z), p(x, f(a), b) \}$

Longest common substring: $p(x,$

The terms in the literals of $S$ that occur immediately to the right of this substring are: $f(x), y, f(a)$

So, $DS = \{ f(x), y, f(a) \}$
Question:

Is \( S = \{ p(x, y), q(x, y) \} \) unifiable?
Example. Trace the algorithm for 
\[ S = \{ p(x, h(x, g(y)), y), p(x, h(a, z), b) \}. \]

1. \( k := 0; \theta_0 := \epsilon. \)
2. \( S\theta_0 = S\epsilon = \{ p(x, h(x, g(y)), y), p(x, h(a, z), b) \} \) is not a singleton; 
   \( D_0 = \{ x, a \}. \)
3. \( \theta_1 := \theta_0 \{ x/a \} = \{ x/a \}; \ k := 1. \)
2. \( S\theta_1 = \{ p(a, h(a, g(y)), y), p(a, h(a, z), b) \} \) is not a singleton; 
   \( D_1 = \{ g(y), z \}. \)
3. \( \theta_2 := \theta_1 \{ z/g(y) \} = \{ x/a, z/g(y) \}; \ k := 2. \)
Example. Trace the algorithm for

\[ S = \{ p(x, h(x, g(y)), y), \ p(x, h(a, z), b) \}. \]

1. \( k := 0; \ \theta_0 := \varepsilon. \)

2. \( S\theta_0 = S\varepsilon = \{p(x, h(x, g(y)), y), p(x, h(a, z), b)\} \) is not a singleton;
   \( D_0 = \{x, a\}. \)

3. \( \theta_1 := \theta_0 \{x/a\} = \{x/a\}; \ k := 1. \)

2. \( S\theta_1 = \{p(a, h(a, g(y)), y), p(a, h(a, z), b)\} \) is not a singleton;
   \( D_1 = \{g(y), z\}. \)

3. \( \theta_2 := \theta_1 \{z/g(y)\} = \{x/a, z/g(y)\}; \ k := 2. \)
Example. Trace the algorithm for

\[ S = \{ p(x, h(x, g(y)), y), \ p(x, h(a, z), b) \}. \]

(conti) …..

- 3. \( \theta_2 := \theta_1[z/g(y)] = \{x/a, z/g(y)\}; \quad k := 2. \)
- 2. \( S\theta_2 = \{ p(a, h(a, g(y)), y), \ \{x/a, z/g(y), b \} \} \) is not a singleton;
  \( D_2 = \{y, b\}. \)
- 3. \( \theta_3 := \theta_2[y/b] = \{x/a, z/g(b), y/b\}; \quad k := 3. \)
- 2. \( S\theta_3 = \{ p(a, h(a, g(b)), b) \} \) is a singleton;
  stop with mgu \( \theta_3 = \{x/a, z/g(b), y/b\}. \)
Resolution Rule \((R)\) (for first-order Logic)

Given the following two clauses

\[ L_1 \lor \ldots \lor L_k \lor C \quad \text{and} \quad \neg M_1 \lor \ldots \lor \neg M_n \lor D, \]

where \(L_i\) and \(M_j\) are atoms and \(C\) and \(D\) are disjunctions of other literals. Assume also that

1. The clauses have distinct sets of variable names.
2. \(\theta\) is the mgu of \(\{L_1, \ldots, L_k, M_1, \ldots, M_n\}\).
3. \(N = L_1 \theta\), where \(\{L_1, \ldots, L_k, M_1, \ldots, M_n\} \theta = \{N\}\).

Then we have:

\[
\frac{L_1 \lor \ldots \lor L_k \lor C, \quad \neg M_1 \lor \ldots \lor \neg M_n \lor D}{(C\theta - N) \lor (D\theta - \neg N)}
\]
Example. Given the two clauses in the following proof segment

\[ k. \quad p(a, y) \lor p(a, z) \lor q(x, y, z) \quad P \]
\[ k + 1. \quad \neg p(w, f(b)) \lor r(w, v, g(w)) \quad P \]

Of the form \( L_1 \lor L_2 \lor C \) and \( \neg M_1 \lor D \).

have distinct sets of variables.

\{p(a, y), p(a, z), p(w, f(b))\} has mgu \( \theta = \{ w/a, y/f(b), z/f(b) \} \).

So, by applying the resolution rule, we get

\[ k + 2. \quad q(x, f(b), f(b)) \lor r(a, v, g(a)) \quad k, k + 1, R, \theta \]
To prove a wff is valid:
1. negate the wff and convert it to clausal form;
2. write down the clauses as premises;
3. use resolution to find a false statement.

Example. Use resolution to prove the following wff is valid.

$$\exists x \ (\forall y \ p(x, y) \lor \forall z \ q(x, z) \rightarrow \forall y \ (p(x, y) \lor q(x, y)))$$

Solution:

Negate the wff:

$$\neg \exists x \ (\forall y \ p(x, y) \lor \forall z \ q(x, z) \rightarrow \forall y \ (p(x, y) \lor q(x, y)))$$
Solution (conti):

Negate the wff:

\[ \neg \exists x ( \forall y p(x, y) \lor \forall z q(x, z) \rightarrow \forall y (p(x, y) \lor q(x, y)). \]

Convert it to clausal form:

\[ \neg \exists x ( \forall y p(x, y) \lor \forall z q(x, z) \rightarrow \forall w (p(x, w) \lor q(x, w)) \] (renamed variables)

\[ \forall x ((\forall y p(x, y) \lor \forall z q(x, z)) \land \exists w (\neg p(x, w) \land \neg q(x, w))) \] (removed \( \rightarrow \) and moved \( \neg \) right)

\[ \forall x \exists w ( (\forall y p(x, y) \lor \forall z q(x, z)) \land \neg p(x, w) \land \neg q(x, w) ) \] (moved \( \exists w \) left)
Solution (conti):

Convert it to clausal form:

\[ \forall x \exists w ( (\forall y p(x, y) \lor \forall z q(x, z)) \land \neg p(x, w) \land \neg q(x, w) ) \]

(moved \( \exists w \) left)

\[ \forall x \exists w \forall y \forall z ( (p(x, y) \lor q(x, z)) \land \neg p(x, w) \land \neg q(x, w) ) \]

(moved \( \forall y \) and \( \forall z \) left)

\[ \forall x \forall y \forall z \{ (p(x, y) \lor q(x, z)) \land \neg p(x, f(x)) \land \neg q(x, f(x)) \} \]

(removed \( \exists w \) by Skolem)

So, set of clauses is

\[ \{ p(x, y) \lor q(x, z) , \neg p(x, f(x)) , \neg q(x, f(x)) \} . \]
Solution (conti):

So the set of clauses is

\[ \{ p(x, y) \lor q(x, z), \neg p(x, f(x)), \neg q(x, f(x)) \} \].

Make each clause a premise

1. \( p(x, y) \lor q(x, z) \)  
2. \( \neg p(u, f(u)) \)  
3. \( \neg q(w, f(w)) \lor \text{False} \)  
4. \( q(u, z) \lor \text{False} \)  
5. []

QED.
4. Computational Logic — Logic programming

Logic Programming

1. What is a logic program?
2. What is a query?
3. How is a query represented?
4. How to see if a query can be answered?
5. How to determine if a query can be inferred from a program?

A logic program is a set of clauses with the restriction that there is exactly one positive literal in each clause. Such clauses are often called definite clauses.

Some are called “Facts” and some are called “definitions”.

Use resolution proof, computation trees

Finite disjunction of literals \((L_1 \lor \ldots \lor L_n)\)
Example. Let \( p(x, y) \) mean \( x \) is a parent of \( y \) and let \( g(x, y) \) mean \( x \) is a grandparent of \( y \).

Some ways to represent definition of the grandparent relation.

First-order logic: \( \forall x \forall y \forall z (p(x, z) \land p(z, y) \rightarrow g(x, y)) \)

First-order clause: \( g(x, y) \lor \neg p(x, z) \lor \neg p(z, y) \)

Logic programming: \( g(x, y) \leftarrow p(x, z), p(z, y) \)

Prolog: \( g(X, Y) :- p(X, Z), p(Z, Y) \)

Definition of \( g \)
A **query** (or, **goal**) is a **question** that asks whether the program infers something: a sequence of one or more atoms. A **question** is whether there is a **substitution** that can be applied to the atoms so that the **resulting atoms** are inferred by the program.

**Example.** Suppose we have the following little logic program.

- `p(a, b)`. // Fact
- `p(b, d)`. // Fact
- `g(x, y) ← p(x, z), p(z, y)`. // Definition
Example. Suppose we have the following little logic program.

- \( p(a, b) \).  // Fact
- \( p(b, d) \).  // Fact
- \( g(x, y) \leftarrow p(x, z), p(z, y) \).  // Definition

Let \( g(a, w) \) be a query. It asks whether \( a \) has a grandchild.

If we let \( \theta = \{ w/d \} \), then \( g(a, w)\theta = g(a, d) \), which says \( a \) has a grandchild \( d \).

This follows from the two program facts \( p(a, b) \) and \( p(b, d) \) and the definition of \( g \).

So \( g(a, d) \) is inferred by the program.
Formal Representation of a Query

From the viewpoint of first-order predicate calculus, a query is the existential closure of a conjunction of one or more atoms.

Example. The query $g(a, w)$ is formally represented by

$$\exists w \ g(a, w).$$

The example program infers $g(a, w)\theta = g(a, d)$ for $\theta = \{w/d\}$.

But EG can be applied to $g(a, d)$ to infer $\exists w \ g(a, w)$. Therefore the program infers $\exists w \ g(a, w)$. 
Example. The query \( g(a, w), p(u, w) \) is formally represented by
\[
\exists w \exists u (g(a, w) \land p(u, w)) .
\]
If we let \( \theta = \{w/d, u/b\} \), then
\[
(g(a, w) \land p(u, w))\theta = g(a, d) \land p(b, d),
\]
which follows from the two example program facts \( p(a, b) \) and \( p(b, d) \) and the definition of \( g \).
So \( g(a, d) \land p(b, d) \) is inferred by the program.
Now apply EG twice to infer \( \exists w \exists u (g(a, w) \land p(u, w)) \).
Therefore the program infers \( \exists w \exists u (g(a, w) \land p(u, w)) \).
To see whether a query can be inferred from a program, a resolution proof is attempted.

The premises are the program clauses together with the negation of the query, which can be written as a clause with only negative literals.

**Example.** Given the query $g(a, w), p(u, w)$. Its formal meaning is $\exists w \exists u (g(a, w) \land p(u, w))$.

So we negate it and convert it to a clause.
Example. Given the query $g(a, w), p(u, w)$. Its formal meaning is

$$\exists w \exists u (g(a, w) \land p(u, w)).$$

So we negate it and convert it to a clause.

Some ways to represent the negation of the query.

**First-order logic:**

$$\neg \exists w \exists u (g(a, w) \land p(u, w)) \equiv \forall w \forall u \neg (g(a, w) \land p(u, w))$$

**First-order clause:**

$$\neg g(a, w) \lor \neg p(u, w)$$

**Logic programming:**

$$\leftarrow g(a, w), p(u, w)$$

**Prolog:**

$$|?- g(a, W), p(U, W)$$
SLD-resolution is a form of resolution used to execute logic programs.

Select the leftmost atom of goal; Linear (each resolvent depends on the previous resolvent); and Definite clauses are the clauses of a logic program.

Example. Given the following little logic program and query:

Logic Programming Syntax

\[ p(a, b). \]
\[ p(b, d). \]
\[ g(x, y) \leftarrow p(x, z), p(z, y). \]

The query:

\[ \leftarrow g(a, w), p(u, w). \]
4. Computational Logic — Logic programming

The **resolution** proof:

<table>
<thead>
<tr>
<th>LPS</th>
<th>FOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. ( p(a, b) )</td>
<td>( p(a, b) )</td>
</tr>
<tr>
<td>2. ( p(b, d) )</td>
<td>( p(b, d) )</td>
</tr>
<tr>
<td>3. ( g(x, y) \leftarrow p(x, z), p(z, y) ) ( g(x, y) \lor \neg p(x, z) \lor \neg p(z, y) )</td>
<td>( P ) (fact)</td>
</tr>
<tr>
<td>4. ( \leftarrow g(a, w), p(u, w) ) ( \neg g(a, w) \lor \neg p(u, w) )</td>
<td>( P )</td>
</tr>
<tr>
<td>5. ( \leftarrow p(a, z), p(z, y), p(u, y) ) ( \neg p(a, z) \lor \neg p(z, y) \lor \neg p(u, y) )</td>
<td>3, 4, R, {x/a, w/y}</td>
</tr>
<tr>
<td>6. ( \leftarrow p(b, y), p(u, y) ) ( \neg p(b, y) \lor \neg p(u, y) )</td>
<td>1, 5, R, {z/b}</td>
</tr>
<tr>
<td>7. ( \leftarrow p(u, d) ) ( \neg p(u, d) )</td>
<td>2, 6, R, {y/d}</td>
</tr>
<tr>
<td>8. [ ]</td>
<td>[ ]</td>
</tr>
<tr>
<td></td>
<td>QED.</td>
</tr>
</tbody>
</table>

University of Kentucky

1/28/2016
The resolution proof:

1. $p(a, b)$                              $p(a, b)$                                        $P$  (fact)
2. $p(b, d)$                               $p(b, d)$                                        $P$  (fact)
3. $g(x, y) \leftarrow p(x, z), p(z, y)$  $g(x, y) \lor \neg p(x, z) \lor \neg p(z, y)$  $P$  (definition)
4. $\leftarrow g(a, w), p(u, w)$          $\neg g(a, w) \lor \neg p(u, w)$                   $P$
5. $\leftarrow p(a, z), p(z, y), p(u, y)$  $\neg p(a, z) \lor \neg p(z, y) \lor \neg p(u, y)$  $3, 4, R, \{x/a, w/y\}$
6. $\leftarrow p(b, y), p(u, y)$           $\neg p(b, y) \lor \neg p(u, y)$                   $1, 5, R, \{z/b\}$
7. $\leftarrow p(u, d)$                    $\neg p(u, d)$                                     $2, 6, R, \{y/d\}$
8. $[]$                                    $[]$  (false)                                      $2, 7, R, \{u/b\}$

QED.  (query satisfied)
The resolution proof:

1. \( p(a, b) \)  \( p(a, b) \)  \( P \) (fact)
2. \( p(b, d) \)  \( p(b, d) \)  \( P \) (fact)
3. \( g(x, y) \leftarrow p(x, z), p(z, y) \)  \( g(x, y) \lor \neg p(x, z) \lor \neg p(z, y) \)  \( P \) (definition)
4. \( \leftarrow g(a, w), p(u, w) \)  \( \neg g(a, w) \lor \neg p(u, w) \)  \( P \)
5. \( \leftarrow p(a, z), p(z, y), p(u, y) \)  \( \neg p(a, z) \lor \neg p(z, y) \lor \neg p(u, y) \)  \( 3, 4, R, \{x/a, w/y\} \)
6. \( \leftarrow p(b, y), p(u, y) \)  \( \neg p(b, y) \lor \neg p(u, y) \)  \( 1, 5, R, \{z/b\} \)
7. \( \leftarrow p(u, d) \)  \( \neg p(u, d) \)  \( 2, 6, R, \{y/d\} \)
8. \( \boxed{} \)  \( \boxed{} \) (false)  \( 2, 7, R, \{u/b\} \)

QED. (query satisfied)
Computation Trees

A computation tree for a query represents all possible ways to satisfy a query. i.e., it represents all possible refutations by resolution.

The root is the query and a child of a node is a resolvant of the node with a program clause.

The mgu's are listed along each branch of the tree.

Computed answers are listed below the leaf of each successful refutation.

Most general unifier

\[
\{ x/c \}
\]

b is the resolvent of a
Example. Given the following logic program and query:

- $p(a, b)$
- $p(c, b)$
- $p(b, d)$
- $g(x, y) \leftarrow p(x, z), p(z, y)$

Query:

$\leftarrow g(w, d)$

The computation tree for the query is shown on the right.
Example. Given the following logic program and query.

- \( p(a) \)
- \( p(x) \leftarrow p(f(x)) \)
- \( p(f(b)) \).

Query:

\( \leftarrow p(y) \)

We have
Example. Given the following logic program and query.

\[ p(a) \]
\[ p(g(x)) \leftarrow p(x) \]
\[ p(b). \]

Query:
\[ \leftarrow p(x) \]

We have
Logic Programming Examples

Example. Let \( \text{ex}(n, m) \) mean the execution of \( p(n), p(n + 1), \ldots, p(m) \). Write a program for the predicate \( \text{ex} \).

Solution:

\[
\text{ex}(n, n) \leftarrow p(n). \\
\text{ex}(n, m) \leftarrow n < m, p(n), k \text{ is } n + 1, \text{ex}(k, m).
\]

\[
\text{ex}(n,m) = \text{if } n=m \text{ then } p(n) \text{ else } p(n), \text{ex}(n+1,m)
\]
4. Computational Logic — Logic programming

**Example.** Let $r$ denote a binary relation. Write logic program to compute the symmetric closure of $r$.

**Solution.** Let $s$ denote the symmetric closure of $r$, use “$s$ contains $(y,x)$ if $(x,y)$ is in $r$” to get

\[
\begin{align*}
&s(x,y) \leftarrow r(x,y) \\
&s(x,y) \leftarrow r(y,x)
\end{align*}
\]
Example. Write a logical program to implement: \( \text{equalLists}(x, y) \) tests if the lists \( x \) and \( y \) are equal.

Solution.

\[
\text{equalLists}(<>, <>).
\]
\[
\text{equalLists}(x::t, x::s) \leftarrow \text{equalLists}(t, s)
\]

\[
\text{equalLists}(x, y) = \text{YES} \text{ if } x=<>; \text{ otherwise } =\text{equalLists}(s, t) \text{ if } x=a::s & y=a::t
\]
End of Computational Logic II
4. Computational Logic — Logic programming

The resolution proof:

1. \( p(a, b) \quad p(a, b) \quad P \) (fact)
2. \( p(b, d) \quad p(b, d) \quad P \) (fact)
3. \( g(x, y) \leftarrow p(x, z), p(z, y) \quad g(x, y) \lor \neg p(x, z) \lor \neg p(z, y) \quad P \) (definition)
4. \( \leftarrow g(a, w), p(u, w) \quad \neg g(a, w) \lor \neg p(u, w) \quad P \)
5. \( \leftarrow p(a, z), p(z, y), p(u, y) \quad \neg p(a, z) \lor \neg p(z, y) \lor \neg p(u, y) \quad 3, 4, R, \{x/a, w/y\} \)
6. \( \leftarrow p(b, y), p(u, y) \quad \neg p(b, y) \lor \neg p(u, y) \quad 1, 5, R, \{z/b\} \)
7. \( \leftarrow p(u, d) \quad \neg p(u, d) \quad 2, 6, R, \{y/d\} \)
8. \( \left[ \right] \quad \left[ \right] \quad 2, 7, R, \{u/b\} \)

\[ \text{QED.} \]
The resolution proof:

1. $p(a, b)$  
2. $p(b, d)$  
3. $g(x, y) \leftarrow p(x, z), p(z, y)$  
4. $\neg g(a, w) \lor \neg p(u, w)$  
5. $\neg p(a, z) \lor \neg p(z, y) \lor \neg p(u, y)$  
6. $\neg p(b, y) \lor \neg p(u, y)$  
7. $\neg p(u, d)$  
8. $[]$  

QED.
4. Computational Logic — Logic programming

The resolution proof:

1. \( p(a, b) \)  \( p(a, b) \) \( P \) (fact)
2. \( p(b, d) \)  \( p(b, d) \) \( P \) (fact)
3. \( g(x, y) \leftarrow p(x, z), p(z, y) \)  \( g(x, y) \lor \neg p(x, z) \lor \neg p(z, y) \) \( P \) (definition)
4. \( \leftarrow g(a, w), p(u, w) \)  \( \neg g(a, w) \lor \neg p(u, w) \) \( P \)
5. \( \leftarrow p(a, z), p(z, y), p(u, y) \)  \( \neg p(a, z) \lor \neg p(z, y) \lor \neg p(u, y) \) \( 3, 4, R, \{x/a, w/y\} \)
6. \( \leftarrow p(b, y), p(u, y) \)  \( \neg p(b, y) \lor \neg p(u, y) \) \( 1, 5, R, \{z/b\} \)
7. \( \leftarrow p(u, d) \)  \( \neg p(u, d) \) \( 2, 6, R, \{y/d\} \)
8. []  [] (false) \( 2, 7, R, \{u/b\} \)

QED. (query satisfied)
4. Computational Logic — Logic programming

The resolution proof:

1. $p(a, b)$  $p(a, b)$  
   $P$ (fact)

2. $p(b, d)$  $p(b, d)$  
   $P$ (fact)

3. $g(x, y) ← p(x, z), p(z, y)$  $g(x, y) ∨ ¬ p(x, z) ∨ ¬ p(z, y)$  
   $P$ (definition)

4. $g(a, w), p(u, w)$  $¬ g(a, w) ∨ ¬ p(u, w)$  
   $P$

5. $p(a, z), p(z, y), p(u, y)$  $¬ p(a, z) ∨ ¬ p(z, y) ∨ ¬ p(u, y)$  
   $3, 4, R, \{x/a, w/y\}$

6. $p(b, y), p(u, y)$  $¬ p(b, y) ∨ ¬ p(u, y)$  
   $1, 5, R, \{z/b\}$

7. $p(u, d)$  $¬ p(u, d)$  
   $2, 6, R, \{y/d\}$

8. $\square$  $\square$ (false)

QED. (query satisfied)
4. Computational Logic – Logic programming

Example. Given the following logic program and query.

- \( p(a) \)
- \( p(g(x)) \leftarrow p(x) \)
- \( p(b) \).

Query:

- \( \leftarrow p(x) \)

We have