Resolution

Logic Lecture 3

Proof by Refutation

Let $A$ be a set of sentences, and let $\alpha$ be a sentence.

$$A \vdash \alpha$$

iff

$A \cup \{\neg \alpha\}$ is unsatisfiable

i.e.

$$A \cup \{\neg \alpha\} \vdash \square$$

$\square$ is the empty clause, or "false."

(To make a clause true, must satisfy at least 1 literal, and $\square$ has no literals.)

Proof by refutation: add $\neg \alpha$ to $A$, and try to prove $\square$ (contradiction).
Propositional Resolution

Complementary pair: \( P \lor q \quad \neg p \lor r \)
\[ q \lor r \]

Example

Given a finite set of clauses, repeatedly choose any two with complementary literals and resolve—add resulting clause.

If \( A \) is set of clauses and resolution derives \( \alpha \), we write \( A \vdash \alpha \), or simply \( A \vdash \alpha \).

Propositional Resolution (Full)

\[ \phi_1 \lor \ldots \lor \phi \lor \ldots \lor \phi_m \]
\[ \psi_1 \lor \ldots \lor \psi \lor \ldots \lor \psi_n \]
\[ \phi_1 \lor \ldots \lor \phi_m \lor \psi_1 \lor \ldots \lor \psi_n \]
**Soundness:** if $A \vdash \alpha$ then $A \models \alpha$.

**Refutation Completeness:** if $A \models \alpha$ then $A \cup \{\neg \alpha\} \vdash 0$.

In reality, for propositional logic we would use DPLL. But it does not scale to predicate logic. This does...

**Resolution for First-Order Logic**

Would like to resolve as follows:

- Recall we assume $X$ is universally quantified outside the entire clause
- $\rho(X) \lor q(X)$
- $\neg \rho(a) \lor r(a)$
- $q(a) \lor r(a)$

But $\rho(X)$ and $\rho(a)$ are not the same, so $\rho(X)$ and $\neg \rho(a)$ cannot be resolved upon as we defined propositional resolution.
Unification (Informal)

Given two (or more) atomic formulas, find a substitution $\Theta$ that will make them all identical.

\[ p(f(X), X) \Theta = p(f(a), a) \]
\[ p(Y, a) \Theta = p(f(a), a) \]

if $\Theta = \{ X \mapsto a, Y \mapsto f(a) \}$

Unifier

Let $E$ be a set of expressions and $\Theta$ be a substitution. If $E\Theta$ is a singleton then $\Theta$ is a unifier of $E$, and we say $\Theta$ unifies $E$. 
More General: Substitution $\theta_1$ is more general than substitution $\theta_2$ (written $\theta_1 \triangleright= \theta_2$) iff:

$$\theta_1 \delta = \theta_2$$

for some substitution $\delta$.

Substitutions ordered by $\triangleright=$ form a "quasi-ordered set" — $\triangleright=$ is reflexive ($\theta \triangleright= \theta$ so $\theta \triangleright= \theta$) and transitive (if $\theta_1 \triangleright= \theta_2$ and $\theta_2 \triangleright= \theta_3$ then $\theta_1 \triangleright= \theta_3$ — $\theta_1 \delta_1 = \theta_2$ and $\theta_2 \delta_2 = \theta_3$ for some $\delta_1$ and $\delta_2$ so $\theta_1 (\delta_1 \delta_2) = \theta_3$).

But not antisymmetric: $\theta_1 \triangleright= \theta_2$ and $\theta_2 \triangleright= \theta_1$ if both are renamings; e.g. $\theta_1 = \{X \leftrightarrow Y\}$ and $\theta_2 = \{Y \leftrightarrow X\}$.

Think of $\triangleright=$ as a partially ordered set on equivalence classes of substitutions.
Examples of $\geq$ over substitutions:

Let $\theta = \{x \mapsto f(x)\}$. Let $d = \{x \mapsto f(f(x))\}$.
Then $\theta \geq d$ since $\theta \theta = d$.

Let $\theta = \{x \mapsto A\}$. Let $d = \{x \mapsto a, A \mapsto a\}$.
Then $\theta \geq d$ since $\theta \theta' = d$ where $\theta' = \{A \mapsto a\}$.

For expressions $e_1$ and $e_2$, $e_1 \geq e_2$ $(e_1$ is more general than $e_2$, $e_2$ is an instance of $e_1$) iff $e_1 \theta = e_2$ for some substitution $\theta$. 
\( \geq \) is a quasi-ordering on expressions.

**Reflexive:** \( e \geq e \) for any expression \( e \).

**Transitive:** If \( e_1 \geq e_2 \) and \( e_2 \geq e_3 \), then by definition \( e_1 \theta \geq e_2 \) and \( e_2 \theta \geq e_3 \) for some \( \theta \), \( d \). Then \( e_1 \theta d = e_3 \), so \( e_1 \geq e_3 \).

**Not Antisymmetric:** If \( e_1 \theta = e_2 \) for some renaming substitution \( \theta \), then \( e_2 \theta'' = e_1 \), where the substitution \( \theta'' \) is the inverse of \( \theta \). We say \( e_1 \) and \( e_2 \) are variants.

\[
p(f(X), X) \theta = p(f(Y), Y) \quad \text{where} \quad \theta = \{X \rightarrow Y\}
\]

\[
p(f(Y), Y) \theta'' = p(f(X), X) \quad \theta'' = \{Y \rightarrow X\}
\]

We say variants are "the same modulo renaming" and view any expression in a set of variants as representative of the equivalence class under \( \geq \). Treating variants as the same lets us view \( \geq \) as a partial order.
Let $\bot$ be a new object such that $e \geq \bot$ for any expression $e$. For any two expressions $e_1$ and $e_2$, let $e$ be $\bot$ if $e_1$ and $e_2$ have no unifier and otherwise let $e$ be $e_1\theta$ where $\theta$ is an mgu of $e_1$ and $e_2$.

Then $e$ is a greatest lower bound (glb) of $e_1$ and $e_2$:

1. $e_1 \geq e$ and $e_2 \geq e$

2. if $e_1 \geq e'$ and $e_2 \geq e'$ then $e \geq e'$

To see (2) note $e_1 \geq e'$ and $e_2 \geq e'$ for unifier $\delta$. Then $\delta\theta = \delta$ for some $\theta'$ since $\theta$ is mgu. Then $e \theta' \geq e'$ so $e \geq e'$.

**Most General Unifier:** For a set of expressions $E$ a substitution $\theta$ is a **most general unifier (mgu)** if:

1. $\theta$ is a unifier of $E$

2. for every other unifier $\theta'$ of $E$, $\theta \geq \theta'$

Because $\geq$ is a quasi-ordering, there are many mgus, all equivalent.
Unification Algorithm

Input: Two expressions $s$ & $s'$.  
Output: success with $mgu$ or failure.  
Let $S$ be $s$ & $s'$.  Repeat until nothing applies:

- Select any $t = \bar{x}$ where $\bar{x}$ not a variable, change to $X = \bar{x}$.
- Erase any $X = \bar{x}$.
- Select any $t = t'$ where neither is a variable.
  - If $t$ and $t'$ are constants:
    - If $t$ and $t'$ differ erase the equation.
    - Else FAIL.
  - Else if $t$ is $p(\bar{a}_1, \ldots, \bar{a}_n)$ and $t'$ is $p(\bar{a}_1', \ldots, \bar{a}_n')$ ($p$ is function symbol or predicate symbol):
    - Replace $t = t'$ by $\bar{a}_i = \bar{a}_i'$, $\bar{a}_j = \bar{a}_j'$.
    - Else FAIL.
  - Else select any $X = \bar{x}$ where $X$ occurs elsewhere and $\bar{x}$ not identical.
    - If $X$ occurs in $t$ then FAIL. (Not done in Alg)
    - Else apply substitution $\{X = \bar{x}\}$ to all other equations (without erasing $X = \bar{x}$).

SUCCEED with output $\{X_1 = \bar{a}_1, \ldots, X_n = \bar{a}_n\}$ where $X_1 = \bar{a}_1, \ldots, X_n = \bar{a}_n$ are the equations left in $S$.

Unification Theorem

Suppose the Unification Algorithm is executed with inputs $e$ and $e'$. If $e$ and $e'$ are not unifiable the algorithm halts with failure. Otherwise the algorithm halts with success, and its output substitution is an $mgu$ of $e$ and $e'$.

Other Results:
- $e$ & $e'$ are unifiable iff $e \varnothing \cap e' \varnothing \neq \emptyset$.
- If $e$ and $e'$ are unifiable with $mgu \neq \emptyset$ then $(e \varnothing)_{\varnothing} = (e' \varnothing)_{\varnothing} = e \varnothing \cap e' \varnothing$.
**Example**

\[ S: \{ p(f(X), Y, X) = p(A, A, f(f(c))) \} \]

\[ \{ f(X) = A, Y = A, X = f(f(c)) \} \]

\[ \{ A = f(X), Y = A, X = f(f(c)) \} \]

\[ \{ A = f(X), Y = f(X), X = f(f(c)) \} \]

\[ \{ A = f(f(f(c))), Y = f(f(f(c))), X = f(f(c)) \} \]

Done—nothing applies.

\[ \emptyset = \{ A \mapsto f(f(f(c))), Y \mapsto f(f(f(c))), X \mapsto f(f(c)) \} \]

---

**Exercise:**

On CS machines, at prompt type

\>` prolog

\>` f(X \ Kleene-bar f(X, a)) \equiv f(b, Y).

\>` unify

\>` loves(mary, X) = loves(Y, Y).

\>` p(X) = p(f(X)).

For this last one, be prepared to kill your Prolog process.
First-order Resolution

If two clauses can be written as:
\[ \alpha_1 \lor \cdots \lor \alpha_k \lor \beta_1 \lor \cdots \lor \beta_j \]
and
\[ \gamma_1 \lor \cdots \lor \gamma_k \lor \delta_1 \lor \cdots \lor \delta_j \]
where all \( \alpha_i \) and \( \beta_i \) are atomic formulas and all \( \gamma_i \) and \( \delta_i \) are literals, such that
\( \{\alpha_{i_1}, \ldots, \alpha_{i_k}, \beta_{j_1}, \ldots, \beta_{j_k}\} \) can be unified with no \( \emptyset \),
then their resolution is defined as
\[ \alpha_{i_1} \lor \cdots \lor \alpha_{i_k} \lor \lor \beta_{j_1} \lor \cdots \lor \beta_{j_k} \]
when resolved on \( \alpha_{i_1}, \ldots, \alpha_{i_k}, \beta_{j_1}, \ldots, \beta_{j_k} \).

Resolution Procedure

Repeatedly draw two clauses, “standardize them apart” (apply a renaming substitution to one so they share no variables), and resolve them if possible. Repeat until derive the empty clause. Note: may resolve a clause with itself.
Example

\[ p(X, X) \lor p(f(Y), Z) \lor \neg q(X, Z) \]
\[ \neg p(w, f(a)) \lor q(w, v) \lor \neg r(w, w) \]
\[ \neg q(f(a), f(a)) \lor q(f(a), v) \lor \neg r(f(a), f(a)) \]

Resolution Given Definite Clauses

It is sufficient to resolve one body literal of a clause with the head literal of another clause.

\[ \text{loves}(X, Y) \leftarrow \text{mother}(X), \text{child-of}(Y, X). \]

\[ \text{mother}(\text{mary}). \]

\[ \text{loves}(X, Y) \leftarrow \text{child-of}(Y, X). \]
**Goals in Logic Programming**

Always an existentially closed conjunction.

$$\exists X (\text{loves}(X, \text{tom}) \land \text{mother}(X))$$

When negated, becomes a universally closed disjunction — a clause. All literals are negative.

$$\forall X (\neg \text{loves}(X, \text{tom}) \lor \neg \text{mother}(X))$$

We add to our program as

$$\neg \text{loves}(X, \text{tom}) \lor \neg \text{mother}(X)$$

Sufficient to resolve goal with a program clause to get a new goal, and repeat with new goal — linear derivation.

---

**Example**

$$\neg \text{loves}(X, \text{tom}), \neg \text{mother}(X)$$

III

$\leftarrow \text{loves}(X, \text{tom}), \text{mother}(X)$

$\neg \text{loves}(X, Y) \leftarrow \text{mother}(X), \text{child}\text{-}of(Y, X)$

$\Theta = \{X = X, Y = \text{tom}\}$

$\leftarrow \text{mother}(X), \text{child}\text{-}of(\text{tom}, X)$

$\Theta = \{X = \text{mary}\}$

$\leftarrow \text{child}\text{-}of(\text{tom}, \text{mary})$

$\Theta = \{\}$