Resolution

Logic Lecture 3

Proof by Refutation

Let A be a set of sentences,

and let α be a sentence.

$$A \models \alpha$$

iff

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

i.e.

$$A \cup \{\neg\alpha\} \models \Box$$

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

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

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

Propositional Resolution

complementary pair

$$\frac{\begin{array}{c} p \vee q \\ \neg p \vee r \end{array}}{q \vee r}$$

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 α, we write $A \vdash_{res} \alpha$, or simply $A \vdash \alpha$.

Propositional Resolution (Full)

$$\frac{\begin{array}{c} \varphi_1 \vee \cdots \vee \gamma \vee \cdots \vee \varphi_m \\ \psi_1 \vee \cdots \vee \neg\gamma \vee \cdots \vee \psi_n \end{array}}{\varphi_1 \vee \cdots \vee \varphi_m \vee \psi_1 \vee \cdots \vee \psi_n}$$

Soundness: if $A \vdash \alpha$ then $A \models \alpha$.

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

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 $\rightarrow$

$$\frac{\begin{array}{c} p(X) \vee q(X) \\ \neg p(a) \vee r(a) \end{array}}{q(a) \vee r(a)}$$

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

Unification (Informal)

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

$$p(f(X), X)\,\theta = p(f(a), a)$$
$$p(Y, a)\,\theta = p(f(a), a)$$

if $\theta = \{X \to a, Y \to f(a)\}$

Unifier

Let E be a set of expressions and θ be a substitution. If $E\theta$ is a singleton then θ is a unifier of E, and we say θ unifies E.

More General: Substitution θ_1 is more general than substitution θ_2 (written $\theta_1 \geq \theta_2$) iff:

$$\theta_1 \delta = \theta_2$$

for some substitution δ.

Substitutions ordered by $\geq$ form a "quasi-ordered set" — $\geq$ is

reflexive ($\theta\varepsilon = \theta$ so $\theta \geq \theta$) and

transitive (if $\theta_1 \geq \theta_2$ and $\theta_2 \geq \theta_3$ then $\theta_1 \geq \theta_3$ — $\theta_1\delta_1 = \theta_2$ and $\theta_2\delta_2 = \theta_3$ for some δ_1 and δ_2 so $\theta_1(\delta_1\delta_2) = \theta_3$)

but not **antisymmetric**: $\theta_1 \geq \theta_2$ and $\theta_2 \geq \theta_1$ if both are renamings, e.g. $\theta_1 = \{X \to Y\}$ and $\theta_2 = \{Y \to X\}$.

Think of $\geq$ as a partially ordered set on equivalence classes of substitutions.

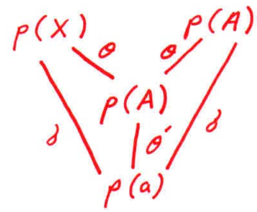
Examples of $\geq$ over substitutions:

Let $\theta = \{X \to f(X)\}$. Let $\delta = \{X \to f(f(X))\}$.

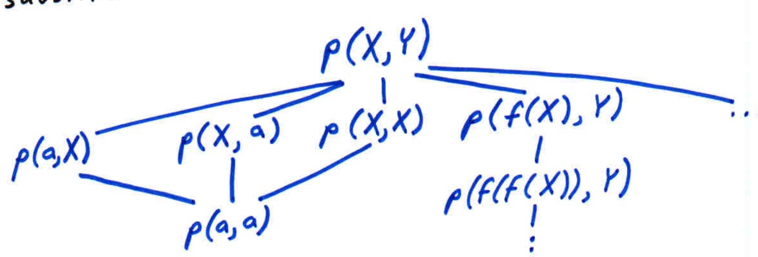
Then $\theta \geq \delta$ since $\theta\theta = \delta$.

Let $\theta = \{X \to A\}$. Let $\delta = \{X \to a, A \to a\}$.

Then $\theta \geq \delta$ since $\theta\theta' = \delta$ where $\theta' = \{A \to a\}$.



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 θ.



$\geq$ is a quasi-ordering on expressions.

Reflexive: $e\varepsilon = e$ for any expression e.

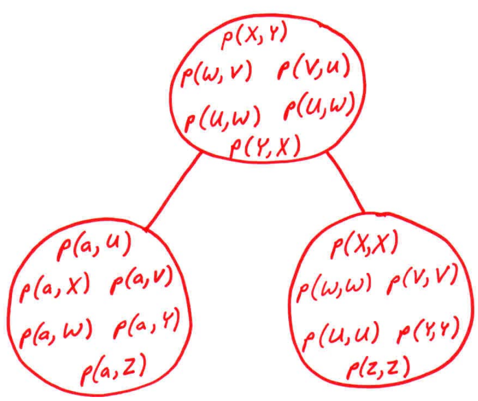
Transitive: If $e_1 \geq e_2$ and $e_2 \geq e_3$ then by definition $e_1\theta = e_2$ and $e_2\delta = e_3$ for some θ, δ. Then $e_1\theta\delta = e_3$, so $e_1 \geq e_3$.

Not Antisymmetric: If $e_1\theta = e_2$ for some renaming substitution θ, then $e_2\theta^{-1} = e_1$ where the substitution θ^{-1} is the inverse of θ. We say e_1 and e_2 are variants.

$p(f(X),X)\theta = p(f(Y),Y)$ where $\theta = \{X \to Y\}$

$p(f(Y),Y)\theta^{-1} = p(f(X),X)$ $\theta^{-1} = \{Y \to X\}$

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 $\perp$ be a new object such that $e \geq \perp$ for any expression e. For any two expressions e_1 and e_2, let e be $\perp$ if e_1 and e_2 have no unifier and otherwise let e be $e_1\theta$ where θ 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 $e_1\delta = e'$ and $e_2\delta = e'$ for unifier δ. Then $\theta\theta' = \delta$ for some θ' since θ is mgu. Then $e\theta' = e'$ so $e \geq e'$.

Most General Unifier: For a set of expressions E a substitution θ is a most general unifier (mgu) if:

(1) θ is a unifier of E

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

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

Unification Algorithm

Input: Two expressions s_1 & s_2.
Output: SUCCESS with mgu θ or FAILURE

Let S be $\{s_1 = s_2\}$. Repeat any until nothing applies:

- Select any $t = X$ where t not a variable, change to $X = t$.
- Erase any $X = X$.
- Select any $t = t'$ where neither is a variable.
  If t and t' are constants:
  If t and t' are same erase the equation.
  Else FAIL.
  Else if t is $p(e_1, \ldots, e_n)$ and t' is $p(e_1', \ldots, e_n')$
  (p is function symbol or predicate symbol):
  Replace $t = t'$ by $e_1 = e_1', \ldots, e_n = e_n'$.
  Else FAIL.
- Select any $X = t$ where X occurs elsewhere and X and t are not identical.
  If X occurs in t then FAIL. (Not done in Prolog)
  Else apply substitution $\{X \to t\}$ to all other equations (without erasing $X = t$).

SUCCEED with output $\{X_1 \to t_1, \ldots, X_n \to t_n\}$ where $X_1 = t_1, \ldots, X_n = t_n$ 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 $e_{gr} \cap e'_{gr} \neq \emptyset$.
- If e and e' are unifiable with mgu θ then $(e\theta)_{gr} = (e'\theta)_{gr} = e_{gr} \cap e'_{gr}$.

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.

$\theta = \{ A \to f(f(f(c))),\ Y \to f(f(f(c))),\ X \to f(f(c)) \}$

Exercise:

On CS machines, at prompt type

```
> prolog
:- f(X,f(X,a)) = 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:

$$\alpha_1 \vee \dots \vee \alpha_k \vee \varphi_1 \vee \dots \vee \varphi_m \qquad k \geq 1$$

and

$$\neg\beta_1 \vee \dots \vee \neg\beta_j \vee \psi_1 \vee \dots \vee \psi_n \qquad j \geq 1$$

where all αs and βs are atomic formulas and all φs and ψs are literals, such that $\{\alpha_1, \dots, \alpha_k, \beta_1, \dots, \beta_j\}$ can be unified with mgu θ, then their resolution is defined as

$$\frac{\begin{array}{l}\alpha_1 \vee \dots \vee \alpha_k \vee \varphi_1 \vee \dots \vee \varphi_m \\ \neg\beta_1 \vee \dots \vee \neg\beta_j \vee \psi_1 \vee \dots \vee \psi_n\end{array}}{\varphi_1\theta \vee \dots \vee \varphi_m\theta \vee \psi_1\theta \vee \dots \vee \psi_n\theta}$$

when resolved on $\alpha_1, \dots, \alpha_k, \neg\beta_1, \dots, \neg\beta_j$.

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

$\gamma = \{X \to f(a), Y \to a, Z \to f(a), W \to f(a)\}$

$$\frac{\begin{array}{c} p(X,X) \vee p(f(Y),Z) \vee \neg q(X,Z) \\ \neg p(W, f(a)) \vee q(W,V) \vee \neg r(W,W) \end{array}}{\neg q(f(a),f(a)) \vee q(f(a),V) \vee \neg r(f(a),f(a))}$$

Resolution Given Definite Clauses

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

$$\frac{\begin{array}{l} loves(X,Y) \leftarrow mother(X), child\text{-}of(Y,X). \\ mother(mary). \end{array}}{loves(X,Y) \leftarrow child\text{-}of(Y,X).}$$

Goals in Logic Programming

Always an existentially closed conjunction.

$$\exists X\,(loves(X, tom) \wedge mother(X))$$

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

$$\forall X\,(\neg loves(X, tom) \vee \neg mother(X))$$

We add to our program as

$$\neg loves(X, tom) \vee \neg mother(X)$$

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

Example

$\neg loves(X, tom), \neg mother(X)$

$\equiv$

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

$\theta = \{X' \to X, Y' \to tom\}$

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

$\theta = \{X \to mary\}$

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

$\theta = \{\}$

$\square$