EECS 3401 — AI and Logic Prog. — Lecture 7

Adapted from slides of Prof. Yves Lesperance

Vitaliy Batusov
vbatusov@cse.yorku.ca

York University

October 5, 2020

Resultion: Unification

- Today: **Unification in FOL Resolution**
- Required reading: Russell & Norvig, Chapters 9.1, 9.2, 9.5

Unification

- **Ground clause** = a clause with no variables
- Finding a pair of complimentary literals $\{p, \neg p\}$ is trivial in ground clauses — syntactical identity suffices
- But what if we have variables in the clauses?

Can these clauses be resolved?

$$(p(john), q(fred), r(X))$$
$$(\neg p(Y), r(susan), r(Y))$$

Unification

- Recall: in clausal form, all variables are universally quantified. So, implicitly, the clause

$$(\neg p(Y), r(susan), r(Y))$$

represents all clauses like

$$(\neg p(fred), r(susan), r(fred))$$
$$(\neg p(john), r(susan), r(john))$$
$$\dots$$

- Thus, there is a "specialization" of this clause that can be resolved with $(p(john), q(fred), r(X))$.

Unification

- We want to be able to match conflicting literals, even if they have variables.
- This matching process automatically determines whether or not there is a "specialization" that matches.
- Don't want to over-specialize!

Consider:

$$(\neg p(X), s(X), q(fred))$$
$$(p(Y), r(Y))$$

Possible resolvants:

- $(s(john), q(fred), r(john)) \qquad \{Y = X, X = john\}$
- $(s(sally), q(fred), r(sally)) \qquad \{Y = X, X = sally\}$
- $(s(X), q(fred), r(X)) \qquad \{Y = X\}$

- The last one is the **most general**, and the first two are specializations of it
- We want to keep the most general clause so that we can use it in future resolution steps

Unification

- **Unification** is a mechanism for finding a "most general" matching

But first,

- A **substitution** is a finite set of equations of the form

$$(V = t)$$

where V is a variable and t is a term *not containing* V. (It might contain other variables)

Substitutions

- We can **apply a substitution** σ to a formula ϕ to obtain a new formula $\phi\sigma$ by simultaneously replacing every variable mentioned in the left-hand side of the substitution by the right-hand side.

  Example:

  $$p(X, g(Y, Z))[X = Y, Y = f(a)] \quad \Rightarrow \quad p(Y, g(f(a), Z))$$

- Note that the substitutions are not applied sequentially, i.e., the first Y is not subsequently replaced by $f(a)$.

Substitutions

We can also compose two substitutions θ and σ to obtain a new substitution $\theta\sigma$.

$$\text{Let } \theta = \{X_1 = s_1, X_2 = s_2, \ldots, X_m = s_m\}$$
$$\sigma = \{Y_1 = t_1, Y_2 = t_2, \ldots, Y_k = t_k\}$$

1. Apply σ to each right-hand side of θ and then add all of the equations of σ.

$$S = \{X_1 = s_1\sigma, X_2 = s_2\sigma, \ldots, X_m = s_m\sigma,$$
$$Y_1 = t_1, Y_2 = t_2, \ldots, Y_k = t_k\}$$

2. Delete from S all identities of the form $V = V$
3. Delete all equations $Y_i = s_i$ where Y_i is equal to one of the X_j in θ

The resulting set S is the composition $\theta\sigma$.

Composition Example

$$\theta = \{X = f(Y), Y = Z\}, \quad \sigma = \{X = a, Y = b, Z = Y\}$$

$$\begin{aligned} S =& \{X = f(Y)\sigma, Y = Z\sigma, X = a, Y = b, Z = Y\} \\ =& \{X = f(b), Y = Y, X = a, Y = b, Z = Y\} \\ =& \{X = f(b), Y = Y, X = a, Y = b, Z = Y\} \\ =& \{X = f(b), X = a, Y = b, Z = Y\} \end{aligned}$$

$$\theta\sigma = \{X = f(b), Y = b, Z = Y\}$$

Substitutions

- The empty substitution $\varepsilon = \{\}$ is also a legal substitution, and it act as an identity under composition
- More importantly, substitutions are **associative** when applied to formulas:

$$(\phi\theta)\sigma = \phi(\theta\sigma)$$

- Composition is simply a way of converting the sequential application of a series of substitutions to a single simultaneous substitution

Unifiers

- A **unifier** of two formulas ϕ and ψ is a substitution σ that makes ϕ and ψ **syntactically identical**
- Not all formulas can be unified — substitutions only affect variables

  Example: the formulas $p(f(X), a)$ and $p(Y, f(w))$ cannot be unified, since there is no way of making $a = f(w)$ with a substitution.

Most General Unifier

- A substitution σ if two formulas ϕ and ψ is a **Most General Unifier (MGU)** if
  1. σ is a unifier
  2. For every other unifier θ of ϕ and ψ there must exist a third substitution λ such that

$$\theta = \sigma\lambda.$$

In other words, σ is a MGU if every other unifier is "more specialized" that σ. The MGU of a pair of formulas ϕ and ψ is unique up to renaming.

Most General Unifier

Consider two formulas:

$$p(f(X), Z), \qquad p(Y, a)$$

- $\sigma = \{Y = f(a), X = a, Z = a\}$ is a unifier:

$$p(f(X), Z)\sigma = p(f(a), a)$$
$$p(Y, a)\sigma = p(f(a), a)$$

  But it is not a MGU.

- $\theta = \{Y = f(X), Z = a\}$ is a MGU:

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

Most General Unifier

- Note: $\sigma = \theta\lambda$ where $\lambda = \{X = a\}$

$$\begin{aligned}\sigma &= \{Y = f(a), X = a, Z = a\} \\ \theta &= \{Y = f(X), Z = a\} \\ \lambda &= \{X = a\}\end{aligned}$$

MGU

$$\theta\lambda = \{Y = f(a), X = a, Z = a\}$$

Most General Unifier

- The MGU is the "least specialized" way of making clauses with universal variables match (syntactically)
- We can find MGUs mechanically
- Intuitively, we line up two formulas and find the first sub-expression where they disagree. The pair of sub-expressions where they **first** disagree is called the **disagreement set**
- The algorithm works by successively fixing disagreements sets until the two formulas become syntactically identical

Most General Unifier

To find the MGU of two formulas ϕ and ψ:

1. $k = 0$; $\sigma_0 = \{\}$, $S_0 = \{\phi, \psi\}$
2. If S_k contains an identical pair of formulas, then stop and return σ_k — this is the MGU
3. Else, find the disagreement set $D_k = \{e_1, e_2\}$ of S_k
4. If e_1 is a variable V and e_2 is a term t not containing V (or vice-versa), then let

$$\sigma_{k+1} = \sigma_k\{V = t\} \qquad \text{(Compose subst.)}$$
$$S_{k+1} = S_k\{V = t\} \qquad \text{(Apply subst.)}$$

   and go back to 2.
5. Else: stop. Formulas ϕ and ψ cannot be unified.

MGU Example 1

$$
\begin{aligned}
S_0 &= \{p(f(a), g(X)); p(Y, Y)\} && k = 0 \\
\sigma_0 &= \{\} \\
D_0 &= \{f(a), Y\} && \color{red}{Y = f(a)}
\end{aligned}
$$

$$
\begin{aligned}
\sigma_1 &= \sigma_0\{Y = f(a)\} = \{Y = f(a)\} && k = 1 \\
S_1 &= \{p(f(a), g(X))\{Y = f(a)\}; \\
&\qquad\qquad p(Y, Y)\{Y = f(a)\}\} \\
&= \{p(f(a), g(X)), p(f(a), f(a))\}
\end{aligned}
$$

$$
D_1 = \{g(X), f(a)\} \qquad \textbf{stop}
$$

MGU Example 2

$$S_0 = \{p(a, X, h(g(Z))); p(Z, h(Y), h(Y))\} \qquad k = 0$$
$$\sigma_0 = \{\}$$
$$D_0 = \{a, Z\} \qquad Z = a$$

$$\sigma_1 = \sigma_0\{Z = a\} = \{Z = a\} \qquad k = 1$$
$$S_1 = \{p(a, X, h(g(a))); p(a, h(Y), h(Y))\}$$
$$D_1 = \{X, h(Y)\} \qquad X = h(Y)$$

$$\sigma_2 = \sigma_1\{X = h(Y)\} = \{Z = a, X = h(Y)\} \qquad k = 2$$
$$S_2 = \{p(a, h(Y), h(g(a))); p(a, h(Y), h(Y))\}$$
$$D_2 = \{g(a), Y\} \qquad Y = g(a)$$

$$\sigma_3 = \sigma_2\{Y = g(a)\} = \{Z = a, X = h(g(a)), Y = g(a)\} \qquad k = 3$$
$$S_3 = \{p(a, h(g(a)), h(g(a))); p(a, h(g(a)), h(g(a)))\}$$

Identical formulas; stop and return σ_3 as the MGU

MGU Example 3

$$S_0 = \{p(X, X); p(Y, f(Y))\} \qquad k = 0$$

$$\sigma_0 = \{\}$$

$$D_0 = \{X, Y\} \qquad X = Y$$

$$\sigma_1 = \sigma_0\{X = Y\} = \{X = Y\} \qquad k = 1$$

$$S_1 = \{p(Y, Y); p(Y, f(Y))\}$$

$$D_1 = \{Y, f(Y)\} \qquad Y = f(Y)$$

Same variable on both sides

Stop; cannot be unified

Non-Ground Resolution

Basic resolution step for non-ground clauses

- If we have two clauses

$$(p, q_1, q_2, \ldots, q_k) \quad \text{and} \quad (\neg m, r_1, r_2, \ldots, r_n)$$

- and if there exists a MGU σ for p and m,
- we infer the new clause

$$(q_1\sigma, \ldots, q_k\sigma, r_1\sigma, \ldots, r_n\sigma)$$

Example of Non-Ground Resolution

- Clauses:

$$(p(X), q(g(X))) \quad \text{and} \quad (r(a), q(Z), \neg p(a))$$

- $\sigma = \{X = a\}$
- Resolve:

$$R[1a, 2c]\{X = a\}(q(g(a)), r(a), q(Z))$$

The notation here is very useful. $R[\cdot, \cdot]$ means a resolution step; $1a$ means the first (a-th) literal in the first (1-st) clause; $2c$ means the third (c-th) literal in the second clause. $\{X = a\}$ is the substitution applied to make the clashing literals identical.

Resolution Proof Example

Consider:
Some patients like all doctors. No patient likes any quack. Therefore, no doctor is a quack.

Resolution Step 1: pick symbols to represent these assertions

- $p(X)$ — X is a patient
- $d(X)$ — X is a doctor
- $q(X)$ — X is a quack
- $l(X, Y)$ — X likes Y

Resolution Proof Example

Resolution Step 2: Convert each assertion to a first-order formula

Some patients like all doctors

$$\exists X(p(X) \wedge \forall Y(d(Y) \rightarrow l(X, Y))) \qquad \text{F1}$$

No patient likes any quack

$$\forall X \forall Y(p(X) \wedge q(Y) \rightarrow \neg l(X, Y)) \qquad \text{F2}$$

Therefore, no doctor is a quack

$$\neg \exists X(d(X) \wedge q(X)) \qquad \text{Query}$$

Resolution Proof Example

Resolution Step 3: Convert to Clausal Form

$(p(a)), (\neg d(X), l(a, X))$	F1
$(\neg p(Y), \neg q(Z), \neg l(Y, Z))$	F2
$(\neg d(V), \neg q(V))$	Query

$(d(b)), (q(b))$	Negation of Query

Resolution Proof Example

Resolution Step 4: Derive an empty clause

$$p(a) \tag{1}$$
$$(\neg d(X), l(a, X)) \tag{2}$$
$$(\neg p(Y), \neg q(Z), \neg l(Y, Z)) \tag{3}$$
$$d(b) \tag{4}$$
$$q(b) \tag{5}$$

$$R[1, 3a]\{Y = a\}(\neg q(Z), \neg l(a, Z)) \tag{6}$$
$$R[2a, 4]\{X = b\} l(a, b) \tag{7}$$
$$R[5, 6a]\{Z = b\} \neg l(a, b) \tag{8}$$
$$R[7, 8]\{\}()$$

Answer Extraction

- The previous example shows how we can answer true-false questions. With a bit more effort we can also answer "fill-in-the-blanks" questions
- As in Prolog, we use free variables in the query where we want the fill-in-the-blanks. We simply need to keep track of the binding that these variables received in proving the query.
  - $parent(art, jon)$ — is *art* one of *jon*'s parents? (Yes/No)
  - $parent(X, jon)$ — who is one of *jon*'s parents? (Fill-in-the-blanks)

Answer Extraction

- A simple bookkeeping device is to use a predicate symbol $answer(X, Y, \ldots)$ to keep track of the bindings automatically
- To answer the query $parent(X, jon)$, we construct the clause

$$(\neg parent(X, jon), answer(X))$$

- Now we perform resolution until we obtain a **clause consisting of only *answer* literals**. (Previously we stopped at empty clauses)

Answer Extraction Example 1

1. $father(art, jon)$
2. $father(bob, kim)$
3. $(\neg father(Y, Z), parent(Y, Z))$
4. $(\neg parent(X, jon), answer(X))$

   Proof:

5. $R[4, 3b]\{Y = X, Z = jon\}(\neg father(X, jon), answer(X))$
6. $R[5, 1]\{X = art\}(answer(art))$
   And we have an explicit answer.

Answer Extraction Example 2

1. $(father(art, jon), father(bob, jon))$
2. $father(bob, kim)$
3. $(\neg father(Y, Z), parent(Y, Z))$
4. $(\neg parent(X, jon), answer(X))$

   Proof:

5. $R[4, 3b]\{Y = X, Z = jon\}(\neg father(X, jon), answer(X))$
6. $R[5, 1a]\{X = art\}(father(bob, jon), answer(art))$
7. $R[6, 3b]\{Y = bob, Z = jon\}(parent(bob, jon), answer(art))$
8. $R[7, 4]\{X = bob\}(answer(bob), answer(art))$
   A disjunctive answer: either *bob* or *art* is a parent of *jon*.

The Prolog search mechanism (without `not` and `!`) is simply an instance of resolution, except

- Clauses are Horn (only one positive literal)
- Prolog uses a specific depth-first strategy when searching for a proof (rules are used first-mentioned-first-used, literals are resolved away left-to-right)

End of Lecture

- Next time: SLDNF Resolution