Resolution Theorem Proving, Part 2

Robert Keller
9 March 2005

MGU Algorithm (Martelli & Montanari)

- Input: Two terms, or two atoms, τ_1, τ_2. Output: Either the most general unifier for τ_1, τ_2, or "not unifiable".
- S := $\{[\tau_1, \tau_2]\}$;
  μ := the empty substitution;
  while(S $\neq \emptyset$)
  remove a pair [L, R] from S; case
  if(L = R) (1)
  do nothing;
  else if($L=f(s_1, s_2, \ldots, s_n)$ and $R=f(t_1, t_2, \ldots, t_n)$) (2)
  S := $S \cup \{[s_1, t_1], [s_2, t_2], \ldots [s_n, t_n]\}$;
  else if(L = x where x is a variable not occurring in R) (3)
  $\mu := \mu \{x \leftarrow R\}$;
  S := applytoallpairs($\{x \leftarrow R\}$, S);
  else if(R = x where x is a variable not occurring in L) (4)
  $\mu := \mu \{x \leftarrow L\}$;
  S := applytoallpairs($\{x \leftarrow L\}$, S);
  else return "not unifiable"; (5)
  return μ as the MGU;

Example

- p(X, f(X)) vs. p(Y, f(Y))
- S := {[p(X, f(X)), p(Y, f(Y))]}
- μ := {}
- Remove [p(X, f(X)), p(Y, f(Y))] case 2
- S := {[X, Y], [f(X), f(Y)]}
- Remove [X, Y] case 3
- μ := {X $\leftarrow$Y}; S := {[f(Y), f(Y)]}
- Remove [f(Y), f(Y)] case 1
- S := {}
- Result: unifiable with mgu {X $\leftarrow$Y}

Example

- p(X, f(X)) vs. p(f(Y), Y)
- S := {[p(X, f(X)), p(f(Y), Y)]}
- μ := {}
- Remove [p(X, f(X)), p(f(Y), Y)] case 2
- S := {[X, f(Y)], [f(X), Y]}
- Remove [X, f(Y)] case 3
- μ := {X $\leftarrow$ f(Y)}; S := {[f(f(Y)), Y]}
- Remove [f(f(Y)), Y] case 5
- Result: not unifiable

Example

- p(X, g(Z), X) vs. p(f(Y), Y, W)
- S := {[p(X, g(Z), X), p(f(Y), Y, W)]}
- μ := {}
- Remove [p(X, g(Z), X), p(f(Y), Y, W)] case 2
- S := {[X, f(Y)], [g(Z), Y], [X, W]}
- Remove [X, f(Y)] case 3
- μ := {X ← f(Y)}; S := {[g(Z), Y], [f(Y), W]}
- Remove [g(Z), Y] case 4
- μ := {X ← f(g(Z)), Y ← g(Z)}; S := {[f(g(Z)), W]}
- Remove [f(g(Z)), W] case 4
- μ := {X ← f(g(Z)), Y ← g(Z), W ← f(g(Z))}; S := {}
- Result: unifiable with
  mgu {X ← f(g(Z)), Y ← g(Z), W ← f(g(Z))}

Try These

τ_1	τ_2	mgu (or not unifiable)
p(X, f(X), d)	p(c, f(c), Y)	
p(f(g(X)), g(Z))	p(f(Y), Y)	
p(f(g(X)), Z)	p(g(Y), Y)	
p(f(g(X)), X)	p(f(g(h(Z))), h(Z))	

Resolving Predicate Calculus Clauses

- Resolvable clauses must contain literals with the same predicate symbol but of opposite sign (one negated, the other not).
- Pick two such literals, one from each clause.
- Rename the variables in one of the clauses so that the two clauses have no variables in common.
- Determine whether the literals are unifiable, with mgu μ. If they are, apply μ to all literals in both clauses. If not, the clauses don't resolve on these particular literals.
- In the modified clauses, remove *all* instances of the modified literals used in unification, and form the disjunction of the remaining literals.

Example Resolving Predicate Clauses

- clause 1: $p(X, g(Z), X) \vee q(X, h(Z))$
- clause 2: $\neg p(f(Y), Y, W) \vee r(f(Y), g(W))$
- No renaming is necessary in this case.
- The first literals unify with mgu
  $\{X \leftarrow f(g(Z)), Y \leftarrow g(Z), W \leftarrow f(g(Z))\}$
- Apply the mgu to both clauses:
- clause 1': $p(f(g(Z)), g(Z), f(g(Z))) \vee q(f(g(Z)), h(Z))$
- clause 2': $\neg p(f(g(Z)), g(Z), f(g(Z))) \vee r(f(g(Z)), g(f(g(Z))))$
- Remove the instances of the unified atoms and form the disjunction.
- Resolvent: $q(f(g(Z)), h(Z)) \vee r(f(g(Z)), g(f(g(Z))))$

Example where there is more than one instance of a literal to remove

- q(b, X) ∨ p(X) ∨ q(b, a)
- ¬q(Y, a) ∨ p(Y)
- unify q(b, X) with ¬q(Y, a)
- mgu is $\{X \leftarrow a, Y \leftarrow b\}$
- Modified clauses:
- q(b, a) ∨ p(a) ∨ q(b, a)
- ¬q(b, a) ∨ p(b)
- There are two instances of q(b, a) in the first clause; both can be removed.
- Resolvent: p(a) ∨ p(b)

Opportunistic Unification of More than Two Literals

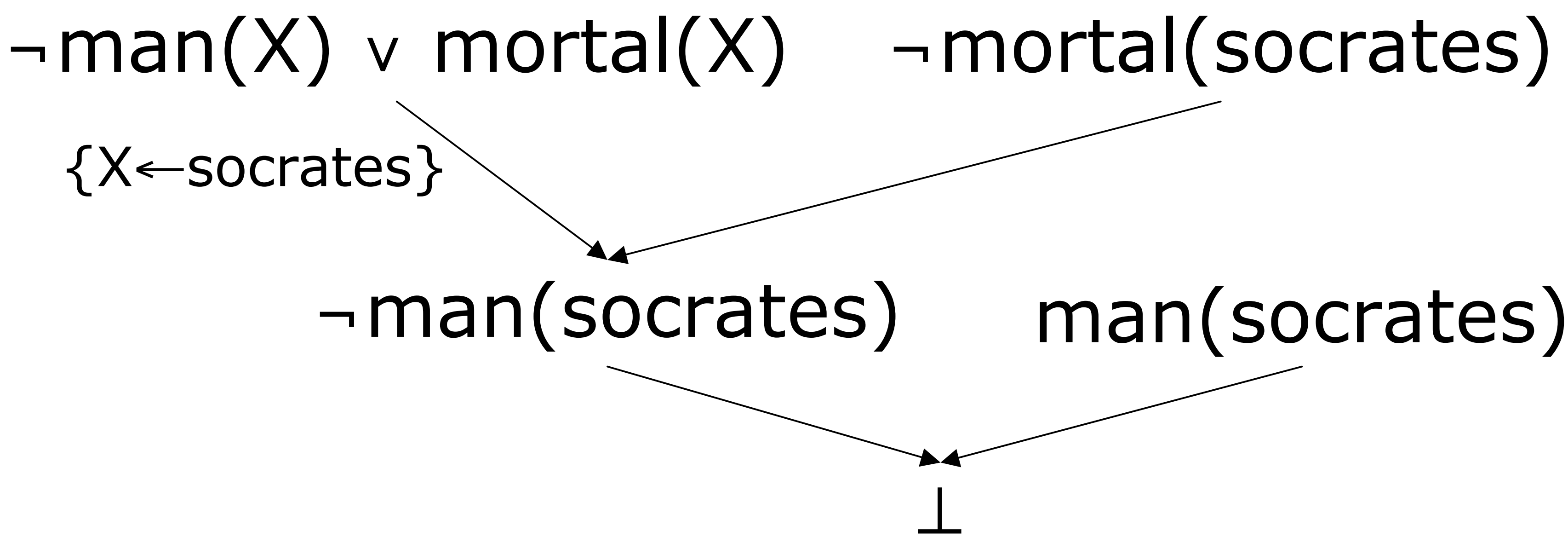
- $p(f(X)) \vee p(Y) \vee \neg q(X)$
- $\neg p(Z) \vee r(W)$
- If we only unify $p(f(X))$ with $p(Z)$, we get
- $p(Y) \vee \neg q(X) \vee r(W)$
- We can then unify $\neg p(Z)$ with $p(Y)$ to get
- $\neg q(X) \vee r(W)$
- Or we could do a 3-way unification in the first place, using the mgu $\{Z \leftarrow f(X), Y \leftarrow f(X)\}$ to get the same result $\neg q(X) \vee r(W)$.
- We will leave the exploration of multi-way unification to you.

Complete Predicate Resolution Process

- The process is the same as for the propositional case, except that we have to rename variables, then unify literals prior to resolution and apply the mgu to all literals in the two clauses, before obtain the resolvent.

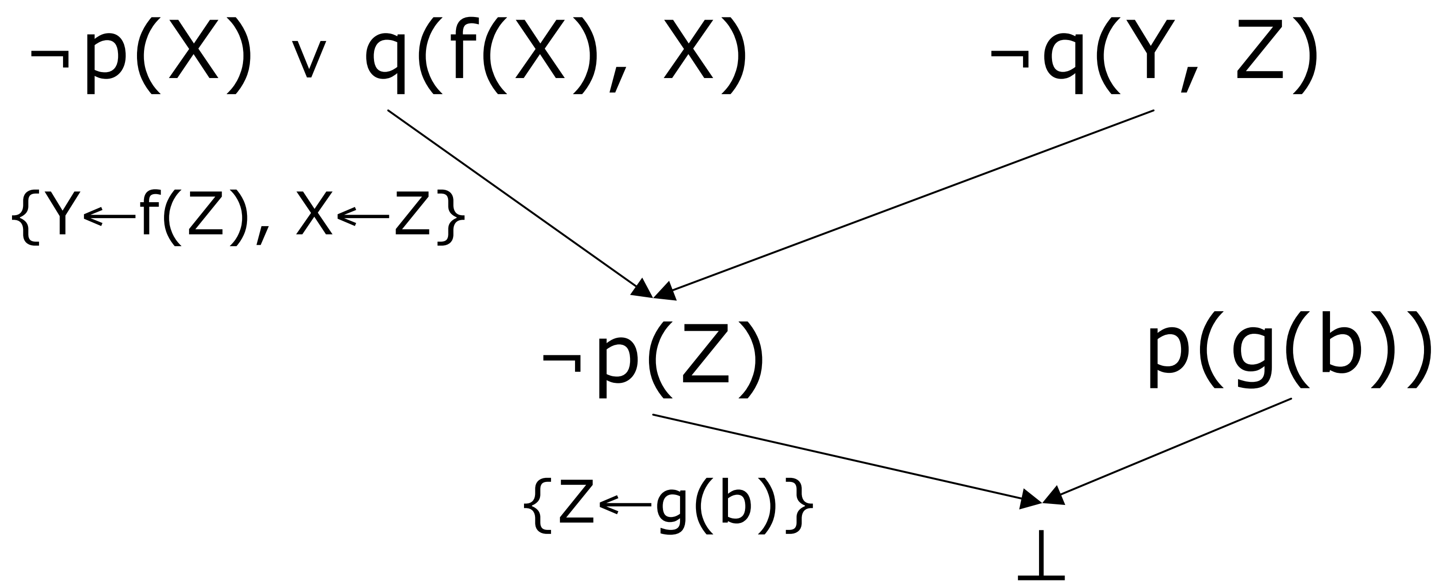
Example of Predicate Resolution

- Clauses:
  - ¬man(X) v mortal(X)
  - man(socrates)
  - ¬mortal(socrates)



Example of Predicate Resolution

- Clauses:
  - $\neg p(X) \vee q(f(X), X))$
  - $p(g(b))$
  - $\neg q(Y, Z)$



Try This Set

1. $\neg e(X) \vee q(X) \vee s(X, f(X))$
2. $\neg e(X) \vee q(X) \vee r(f(X))$
3. $p(a)$
4. $e(a)$
5. $\neg s(a, Y) \vee p(Y)$
6. $\neg p(X) \vee \neg q(X)$
7. $\neg p(X) \vee \neg r(X)$

Clausal Form for Predicate Logic

- Often, we'll want to prove a sequent of the form
  - ∀x ∀y (…)
  - ∀x ∀y (…)

    |− …
  - For premises of the form ∀x ∀y (…) where … has no quantifiers, we can just drop the quantifiers.

  - We need to **negate** the conclusion.

Mushroom Example

1. Every fungus is a mushroom or a toadstool.
2. Every boletus is a fungus.
3. All toadstools are poisonous.
4. No boletus is a mushroom.
5. This thing is a boletus.
6. Therefore: This thing is poisonous.

Mushroom Example

1. $\forall X$ fungus(x) $\rightarrow$ (mushroom(X) $\vee$ toadstool(X))
2. $\forall X$ boletus(X) $\rightarrow$ fungus(X)
3. $\forall X$ toadstool(X) $\rightarrow$ poisonous(X)
4. $\forall X$ boletus(X) $\rightarrow$ $\neg$mushroom(X)
5. boletus(b) (b is a constant standing for "this thing".)
6. Therefore: poisonous(b)

Mushroom Clauses

1. $\neg$fungus(X) $\vee$ mushroom(X) $\vee$ toadstool(X)
2. $\neg$boletus(X) $\vee$ fungus(X)
3. $\neg$toadstool(X) $\vee$ poisonous(X)
4. $\neg$boletus(X) $\vee$ $\neg$mushroom(X)
5. boletus(b)
6. $\neg$poisonous(b) (negated conclusion)

Using Otter

- Otter is a sophisticated resolution theorem prover developed at Argonne National Laboratory.
- It will accept input in either clausal form, or non-clausal form.
- There is a version installed on turing.
- There is an on-line version Otter-λ which has enhancements for the calculus:
  http://mh215a.cs.sjsu.edu/otter_simple.php

Mushroom Clauses in Otter

```
set(auto).
list(usable).

-fungus(x) | mushroom(x) | toadstool(x).

-boletus(x) | fungus(x).

-toadstool(x) | poisonous(x).

-boletus(x) | -mushroom(x).

 boletus(b).

-poisonous(b).

end_of_list.
```

Otter's Output for Mushrooms

```
---------------- PROOF ----------------

1 [] -fungus(x)|mushroom(x)|toadstool(x).
2 [] -boletus(x)|fungus(x).
3 [] -toadstool(x)|poisonous(x).
4 [] -boletus(x)| -mushroom(x).
5 [] -poisonous(b).
6 [] boletus(b).
7 [hyper,6,2] fungus(b).
8 [hyper,7,1] mushroom(b)|toadstool(b).
9 [hyper,8,3,unit_del,5] mushroom(b).
10 [hyper,9,4,6] $F.

------------ end of proof -------------
```

[...] indicates rule name and formulas used

$F indicates the null clause.

Clausal Form for Predicate Logic

- Often, we'll want to prove a sequent of the form
  - ∀x ∀y (…)
  - ∀x ∀y (…)

    |– ∀x ∀y (…)

  - For premises of the form ∀x ∀y (…) where … has no quantifiers, we can just drop the quantifiers.

  - We need to **negate** the conclusion, so that will become
    ¬∀x ∀y (…) which is equivalent to ∃x ∃y ¬(…).

    We can't simply drop the quantifiers in this case!!

Clausal Form for Predicate Logic

- Consider the sequent

  $\forall x\ p(x)\ |\!-\ \forall x\ p(x)$

- The premise translates to a clause
  $p(x)$

- The conclusion is negated to become $\exists x\ \neg p(x)$.

- How do we handle this?

Skolem Functions to the Rescue!

- To get rid of the quantifier in

  $\exists x \neg p(x)$

  we use a trick:

  Create a **new function symbol**, in this case 0-ary, say b(), and replace x with that:

  $\neg p(b())$

- Some thought will show that there is an interpretation that induces true in the latter iff there is one that induces true in the formula.

- This is because we get to pick the value for b() in the interpretation, just as get to pick the value for x.

Clausal Form for Predicate Logic

- Consider the sequent

  $\forall x\ p(x) \vdash \forall x\ p(x)$

- The premise translates to a clause

  $p(x)$

- The conclusion translates to a clause

  $\neg p(b())$

- We are good to go!

- Resolution produces $\bot$ in 1 step.

Another Example

- Consider the sequent

$$\exists x\, \forall y\, p(x, y) \vdash \forall y\, \exists x\, p(x, y)$$

- Premise clause:

$$p(b(), y)$$

- Conclusion clause:

$$\neg p(x, c())$$

Skolem Functions for the General Case

- ∀x ∀y … ∃v …
- v is replaced with f(x, y, …)
- f is a **new function symbol**, the arguments of which are the ∀ quantified variables on the left.

- The rationale here is the “the v” that exists **depends on** x, y, … .

- Again, there is an interpretation satisfying the original formula iff there is an interpretation satisfying the revised formula.

- We love Skolem!

Skolem Example with Argument

- Consider the sequent

$$\forall x\ \exists y\ p(x, y) \mid\!- \forall y\ \exists x\ p(y, x)$$

- Premise clause:

$$p(x, f(x))$$

- Conclusion clause:

$$\neg p(b(), x)$$

- Remember that we have to rename when resolving.

How to get a clause form in general?

- First convert the formula into “**prenex form**” (all quantifiers are outside on the left). [The parts of this form are the “prefix” and the “matrix”.]

- Skolemize ∃ quantified variables.

- Drop ∀ quantifiers.

- Convert the resulting matrix to CNF.

Conversion to Prenex Form

- Replace all connectives other than $\wedge \vee \neg$ with their counterparts.
- Push negation signs inward toward atoms.
- Pull quantifiers to the outside using the rules on the next page.

Prenex Quantifier Rules

- Here Q means either quantifier: ∀ or ∃.
- F[x] means that x occurs in F. If we don't write [x], then it doesn't.
- ⇒ means "replace with".

1. $(Qx\ F[x]) \wedge G \Rightarrow (Qx\ F[x] \wedge G)$
2. $(Qx\ F[x]) \vee G \Rightarrow (Qx\ F[x] \vee G)$

3. $(\forall x\ F[x]) \wedge (\forall x\ G[x]) \Rightarrow \forall x\ (F[x] \wedge G[x])$
4. $(\exists x\ F[x]) \vee (\exists x\ G[x]) \Rightarrow \exists x\ (F[x] \vee G[x])$

5. $(Q_1 x\ F[x]) \wedge (Q_2 x\ G[x]) \Rightarrow Q_1 x\ Q_2 y\ (F[x] \wedge G[y])$
6. $(Q_1 x\ F[x]) \vee (Q_2 x\ G[x]) \Rightarrow Q_1 x\ Q_2 y\ (F[x] \vee G[y])$

Example of Prenex Conversion

- $\forall x\ \forall y((\exists z(p(x, z) \wedge p(y, z)) \rightarrow \exists u\ q(x, y, u))$
- $\forall x\ \forall y(\neg(\exists z(p(x, z) \wedge p(y, z)) \vee \exists u\ q(x, y, u))$
- $\forall x\ \forall y((\forall z\ \neg(p(x, z) \wedge p(y, z)) \vee \exists u\ q(x, y, u))$
- $\forall x\ \forall y((\forall z\ (\neg p(x, z) \vee \neg p(y, z)) \vee \exists u\ q(x, y, u))$
- $\forall x\ \forall y\ \forall z(\neg p(x, z) \vee \neg p(y, z) \vee \exists u\ q(x, y, u))$
- $\underbrace{\forall x\ \forall y\ \forall z\ \exists u}_{\text{prefix}}\underbrace{(\neg p(x, z) \vee \neg p(y, z) \vee q(x, y, u))}_{\text{matrix}}$

Completion of Conversion

- $\forall x\ \forall y\ \forall z\ \exists u(\neg p(x, z) \vee \neg p(y, z) \vee q(x, y, u))$
- Skolemize $\exists u$ and drop $\forall x\ \forall y\ \forall z$
- $\neg p(x, z) \vee \neg p(y, z) \vee q(x, y, f(x, y, z))$

Example: Group Theory Clauses

- $\forall x\ \forall y\ \forall x\ e(f(x, f(y, z)), f(f(x, y), z))$
  becomes
  $e(f(x, f(y, z)), f(f(x, y), z))$

- $\forall x\ e(f(x, u), x)$
  becomes
  $e(f(x, u), x)$

- $\forall x\ e(f(x, i(x)), u)$
  becomes
  $e(f(x, i(x)), u)$

Example: Equality Theory Clauses

- We need to axiomatize equality predicate e, e.g.

- $\forall x\ e(x, x)$
  becomes
  $e(x, x)$

- $\forall x\ \forall y\ \forall u\ \forall v\ (e(x, y) \wedge e(v, w)) \rightarrow e(f(x, v), f(y, w))$
  becomes
  $\neg e(x, y) \vee \neg e(u, v) \vee e(f(x, v), f(y, w))$

- $\forall x\ \forall y\ e(x, y) \rightarrow e(y, x)$
  becomes
  $\neg e(x, y) \vee e(y, x)$

  etc.

Example of Group Theory Clauses with Negated Conclusion

1. $e(f(x, f(y, z)), f(f(x, y), z))$
2. $e(f(x, u), x)$
3. $e(f(x, i(x)), u)$
4. $e(x, x)$
5. $\neg e(x, y) \vee \neg e(v, w) \vee e(f(x, v), f(y, w))$
6. $\neg e(x, y) \vee e(y, x)$
7. $\neg e(x, y) \vee \neg e(y, z) \vee e(x, z)$

8. $\neg e(i(i(b)), b)$

This is to show that $\forall x \; e(i(i(x)), x)$.