Artificial Intelligence -- 15-381
Unification and Resolution in FOL

Jaime Carbonell
8-February-2000

OUTLINE:

- Q/A by matching (partial unification)
- Full term unification in FOL
- FOL in clause form (with Skolemization)
- FOL Robinson Resolution
- Uses and limitations of Resolution

Answering Questions by Matching

Data Base of Ground Instances

```
Person(John)
Person(Mary)
Dog(Fido)
Likes(John,Mary)
Not(Likes(Mary,John))
Likes(Mary,Fido)
Likes(Fido,Mary)
Likes(Fido,John)
Likes(Fido,Fido)
Owners2(Fido,John,Mary)
```

Possible queries

Person(x)
Likes(Mary,x)
Likes(x,x)
Likes(x,y)
Not(Likes(z,John))
Owners(John,x,y)

Simple Matching

- Token match on predicates
- Variables only for query arguments
  => const-const or const-variable match
- Zero, one or multiple answers

Full Unification

Variable-Variable Matching

P(x)	P(y)
Q(x,x)	Q(y,z)
P(f(x),z)	P(y,Fido)

Unifiers: Variable Substitutions

P(x)	P(y)	{y/x}
Q(x,x)	Q(y,z)	{y/x,z/x}
P(f(x),z)	P(y,Fido)	{y/f(x),z/Fido}

Consistent Variable assignments

P(Mary,John)	P(y,y)	#
R(x,y,y)	R(y,y,z)	{y\z,x\y}
W(P(x),y,z)	W(Q(x),y,Fido)	#

Advantages of full unification

- Query and data => both full FOL
- Permits full FOL Resolution (next)

Unification

$Q(x)$
$P(y)$ $\rightarrow$ FAIL

$P(x)$
$P(y)$ $\rightarrow$ x/y

$P(Marcus)$
$P(y)$ $\rightarrow$ Marcus/y

$P(Marcus)$
$P(Julius)$ $\rightarrow$ FAIL

$P(x, x)$
$P(y, z)$ $\rightarrow$ (y/x)
$P(y, y)$ $\rightarrow$ $(z/y)(y/x)$

Finding General Substitutions

Given:

$hate(x, y)$
$hate(Marcus, z)$

We could produce:

$(Marcus/x, z/y)$
$(Marcus/x, y/z)$
$(Marcus/x, Caesar/y, Caesar/z)$
$(Marcus/x, Polonius/y, Polonius/z)$

Algorithm: Unify(L1,L2)

1. If $L1$ or $L2$ is a variable or constant, then:
   (a) If $L1$ and $L2$ are identical, then return NIL.
   (b) Else if $L1$ is a variable, then if $L1$ occurs in $L2$ then return FAIL, else return $\{(L2/L1)\}$.
   (c) Else if $L2$ is a variable then if $L2$ occurs in $L1$ then return FAIL, else return $\{(L1/L2)\}$.
   (d) Else return FAIL.
2. If the initial predicate symbols in $L1$ and $L2$ are not identical, then return FAIL.
3. If $L1$ and $L2$ have a different number of arguments, then return FAIL.
4. Set *SUBST* to NIL.
5. For $i \leftarrow 1$ to number of arguments in $L1$:
   (a) Call Unify with the ith argument of $L1$ and the ith argument of $L2$, putting result in S.
   (b) If S = FAIL then return FAIL.
   (c) If S is not equal to NIL then:
      i. Apply S to the remainder of both $L1$ and $L2$.
      ii. *SUBST* := APPEND(S, *SUBST*).
6. Return *SUBST*.

A Predicate Logic Example

1. Marcus was a man.
   $man(Marcus)$
2. Marcus was a Pompeian.
   $Pompeian(Marcus)$
3. All Pompeians were Romans.
   $\forall x : Pompeian(x) \rightarrow Roman(x)$
4. Caesar was a ruler.
   $ruler(Caesar)$
5. All Romans were either loyal to Caesar or hated him.
   $\forall x : Roman(x) \rightarrow loyalto(x, Caesar) \vee hate(x, Caesar)$
6. Everyone is loyal to someone.
   $\forall x : \exists y : loyalto(x, y)$
7. People only try to assassinate rulers they aren't loyal to.
   $\forall x : \forall y : person(x) \wedge ruler(y) \wedge tryassassinate(x, y)$
   $\rightarrow \neg loyalto(x, y)$
8. Marcus tried to assassinate Caesar.
   $tryassassinate(Marcus, Caesar)$

---

9. All men are people.
   $\forall x : man(x) \rightarrow person(x)$

Conversion to Clause Form

Problem:

$$\forall x : [Roman(x) \wedge know(x, Marcus)] \rightarrow$$
$$[hate(x, Caesar) \vee$$
$$(\forall y : \exists z : hate(y, z) \rightarrow thinkcrazy(x, y))]$$

Solution:

- Flatten
- Separate out quantifiers

Conjunctive Normal Form:

$$\neg Roman(x) \vee \neg know(x, Marcus) \vee$$
$$hate(x, Caesar) \vee \neg hate(y, z) \vee thinkcrazy(x, z)$$

Clause Form:

- Conjunctive normal form
- No instances of $\wedge$

Algorithm: Convert to Clause Form

1. Eliminate $\rightarrow$, using: $a \rightarrow b = \neg a \vee b$.
2. Reduce the scope of each $\neg$ to a single term, using:
   - $\neg(\neg p) = p$
   - deMorgan's laws: $\neg(a \wedge b) = \neg a \vee \neg b$, $\neg(a \vee b) = \neg a \wedge \neg b$
   - $\neg \forall x : P(x) = \exists x : \neg P(x)$
   - $\neg \exists x : P(x) = \forall x : \neg P(x)$
3. Standardize variables.
4. Move all quantifiers to the left of the formula without changing their relative order.
5. Eliminate existential quantifiers by inserting Skolem functions.
6. Drop the prefix.
7. Convert the matrix into a conjunction of disjuncts, using associativity and distributivity.
8. Create a separate clause for each conjunct.
9. Standardize apart the variables in the set of clauses generated in step 8, using the fact that
   $(\forall x : P(x) \wedge Q(x)) = \forall x : P(x) \wedge \forall x : Q(x)$

Skolem Functions in FOL

Objective

- Want all variables universally quantified
- Notational variant of FOL w/o existentials
- Retain implicitly full FOL expressiveness

Skolem's Theorem (simplified)

> Every existentially quantified variable can be replaced by a unique ***Skolem function*** whose arguments are all the universally quantified variables on which the existential depends

Examples

"Everybody likes something"
All(x)Exists(y)[Person(x) & Likes(x,y)]
All(x)[Person(x) & Likes(x, S1(x))]
Where S1(x) = "that which x likes"

"Every philosopher writes at least one book"
A(x)E(y)[(Philosopher(x) & Book(y)) => Write(x,y)]
A(x)[(Philosopher(x) & Book(S2(x))) => Write(x,S2(y))]

Examples of Conversion to Clause Form

Example:

$$\forall x : [Roman(x) \wedge know(x, Marcus)] \rightarrow [hate(x, Caesar) \vee (\forall y : \exists z : hate(y, z) \rightarrow thinkcrazy(x, y))]$$

. Eliminate $\rightarrow$.

$$\forall x : \neg[Roman(x) \wedge know(x, Marcus)] \vee [hate(x, Caesar) \vee (\forall y : \neg(\exists z : hate(y, z)) \vee thinkcrazy(x, y))]$$

. Reduce scope of $\neg$.

$$\forall x : [\neg Roman(x) \vee \neg know(x, Marcus)] \vee [hate(x, Caesar) \vee (\forall y : \forall z : \neg hate(y, z) \vee thinkcrazy(x, y))]$$

. Standardize variables.

$\forall x : P(x) \vee \forall x : Q(x)$

will be converted to

$\forall x : P(x) \vee \forall y : Q(y)$

. Move quantifiers.

$$\forall x : \forall y : \forall z : [\neg Roman(x) \vee \neg know(x, Marcus)] \vee [hate(x, Caesar) \vee (\neg hate(y, z) \vee thinkcrazy(x, y))]$$

Examples of Conversion to Clause Form

5. Eliminate existential quantifiers.

   $\exists y : President(y)$

   will be converted to

   $President(S1)$

   While

   $\forall x : \exists y : father\text{-}of(y, x)$

   will be converted to

   $\forall x : father\text{-}of(S2(x), x))$

6. Drop the prefix.

   $[\neg Roman(x) \vee \neg know(x, Marcus)] \vee$
   $[hate(x, Caesar) \vee (\neg hate(y, z) \vee thinkcrazy(x, y))]$

7. Convert to a conjunction of disjuncts.

   $\neg Roman(x) \vee \neg know(x, Marcus) \vee$
   $hate(x, Caesar) \vee \neg hate(y, z) \vee thinkcrazy(x, y)$

The Basis of Resolution

Given:

winter $\vee$ *summer*
$\neg$*winter* $\vee$ *cold*

We can conclude:

summer $\vee$ *cold*

Herbrand's Theorem

To show that a set of clauses S is unsatisfiable, it is necessary to consider only interpretations over a particular set, called the *Herbrand universe* of S.

A set of clauses S is unsatisfiable if and only if a finite subset of ground instances (in which all bound variables have had a value substituted for them) of S is unsatisfiable.

Algorithm: Propositional Resolution

1. Convert all the propositions of F to clause form.

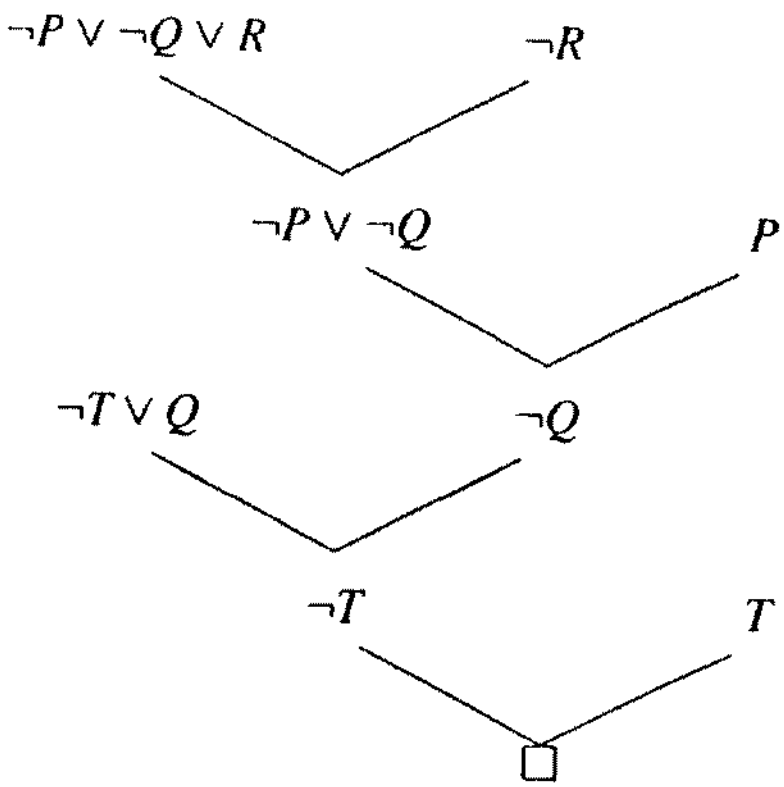
2. Negate P and convert the result to clause form. Add it to the set of clauses obtained in step 1.

3. Repeat until either a contradiction is found or no progress can be made:
   (a) Select two clauses. Call these the parent clauses.
   (b) Resolve them together. The *resolvent* will be the disjunction of all of the literals of both of the parent clauses with the following exception: If there are any pairs of literals L and $\neg L$ such that one of the parent clauses contains L and the other contains $\neg L$, then select one such pair and eliminate both L and $\neg L$ from the resolvent.
   (c) If the resolvent is the empty clause, then a contradiction has been found. If it is not, then add it to the set of clauses available to the procedure.

A Few Facts in Propositional Logic

Given Axioms	Clause Form	
P	P	(1)
$(P \wedge Q) \rightarrow R$	$\neg P \vee \neg Q \vee R$	(2)
$(S \vee T) \rightarrow Q$	$\neg S \vee Q$	(3)
	$\neg T \vee Q$	(4)
T	T	(5)

Resolution in Propositional Logic



Algorithm: Resolution

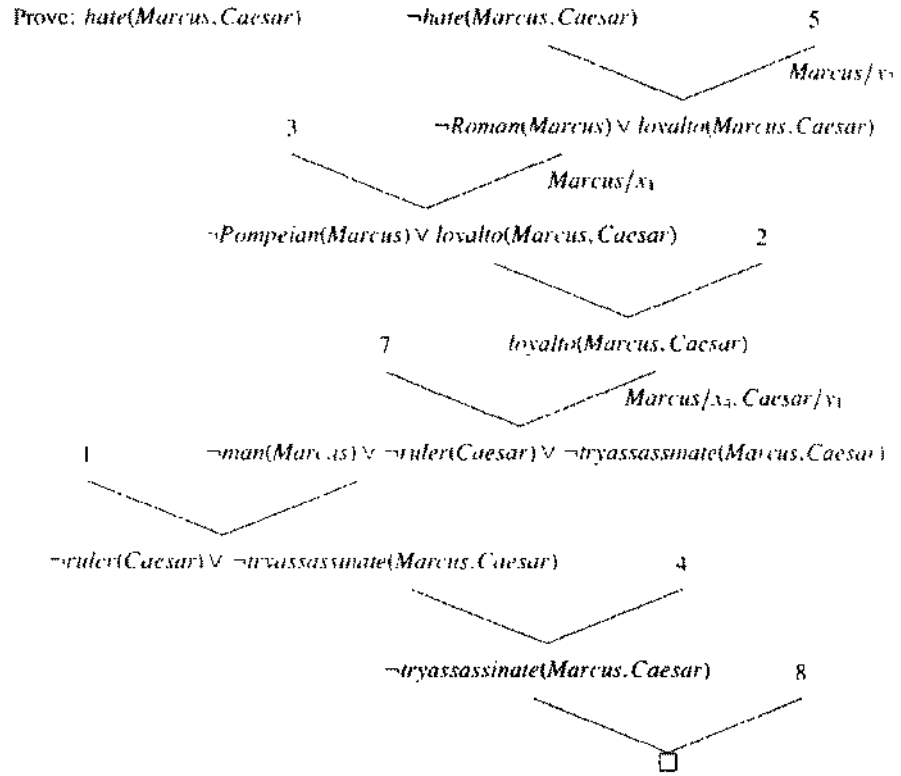
1. Convert all the statements of F to clause form.
2. Negate P and convert the result to clause form. Add it to the set of clauses obtained in 1.
3. Repeat until either a contradiction is found, no progress can be made, or a predetermined amount of effort has been expended.
   (a) Select two clauses. Call these the parent clauses.
   (b) Resolve them together. The resolvent will be the disjunction of all the literals of both parent clauses with appropriate substitutions performed and with the following exception: If there is one pair of literals $T1$ and $\neg T2$ such that one of the parent clauses contains $T1$ and the other contains $\neg T2$ and if $T1$ and $T2$ are unifiable, then neither $T1$ nor $\neg T2$ should appear in the resolvent. If there is more than one pair of complementary literals, only one pair should be omitted from the resolvent.
   (c) If the resolvent is the empty clause, then a contradiction has been found. If it is not, then add it to the set of clauses available to the procedure.

A Resolution Proof

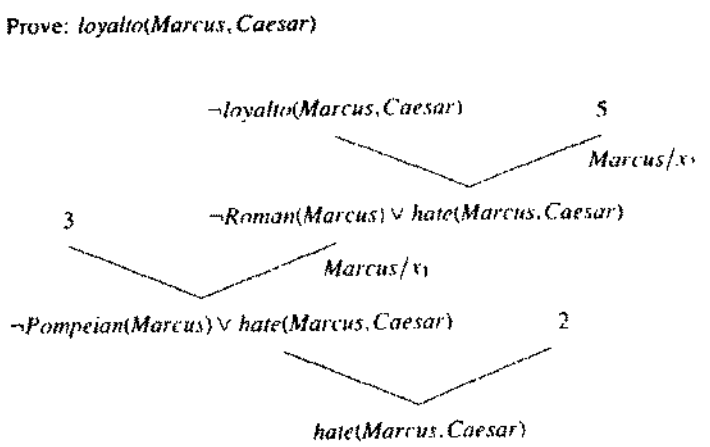
Axioms in clause form:

1. $man(Marcus)$
2. $Pompeian(Marcus)$
3. $\neg Pompeian(x_1) \vee Roman(x_1)$
4. $ruler(Caesar)$
5. $\neg Roman(x_2) \vee loyalto(x_2, Caesar) \vee hate(x_2, Caesar)$
6. $loyalto(x_3, f1(x_3))$
7. $\neg man(x_4) \vee \neg ruler(y_1) \vee \neg tryassassinate(x_4, y_1) \vee loyalto(x_4, y_1)$
8. $tryassassinate(Marcus, Caesar)$

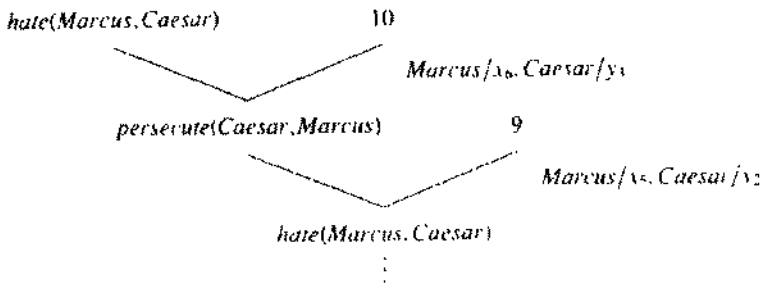
(*a*)



An Unsuccessful Attempt at Resolution



(a)



(b)

Using Resolution with Equality and Reduce

Axioms in clause form:

1. $man(Marcus)$
2. $Pompeian(Marcus)$
3. $born(Marcus, 40)$
4. $\neg man(x_1) \vee mortal(x_1)$
5. $\neg Pompeian(x_2) \vee died(x_2, 79)$
6. $erupted(volcano, 79)$
7. $\neg mortal(x_3) \vee \neg born(x_3, t_1) \vee \neg gt(t_2 - t_1, 150) \vee dead(x_3, t_2)$
8. $now = 1991$

9a. $\neg alive(x_4, t_3) \vee \neg dead(x_4, t_3)$

9b. $dead(x_5, t_4) \vee alive(x_5, t_4)$

10. $\neg died(x_6, t_5) \vee \neg gt(t_6, t_5) \vee dead(x_6, t_6)$

Prove: $\neg alive(Marcus, now)$