Resolution

Predicate Logic

Facts and Knowledge are represented as
Logical Statements

New or Extended Knowledge is gained by
Automatic Theorem Proving

OTTER
System for automatic theorem proving

OTTER

Organized
Techniques for
Theorem Proving and
Effective
Research

Resolution Style Prover
for
First Order Logic
(with equality)

Will be used here to illustrate the basic principles of resolution and automatic theorem proving

Its abilities extend much further than what will be shown here

Terminology

Predicate
Logical Property
n-ary symbol denoting a relation
person(marcus)
born(marcus,40)

Literal
Either a predicate $P(x, y, ...)$
or
not predicate $(\neg P(x, y, ...))$

Clause
Literals separated by **OR**
$P \wedge Q(x) \wedge \neg R(x, y)$

OTTER clause list

Negative Literal
$\neg P(x, y, ...)$

Positive Literal
P(x,y,...)

Positive Clause
Only Positive literals

Negative Clause
Only Negative Literals

Mixed Clause
At least one positive and at least one negative literal

Unit Clause
Only one Literal

OTTER clause list

```
list(usable).
pompeian(marcus).
born(marcus,40).
-alive(marcus,1995).
end_of_list.
```

OTTER Formulas

Atoms

Atoms are Formulas

implies and iff

If F and G are formulas then

$\underline{(F <-> G)}$

$\underline{(F-> G)}$

AND and OR

If $F_1, F_2, ..., F_n$ are formulas, then

$\underline{(F_1 \mid F_2 \mid ... \mid F_n)}$

$\underline{(F_1 \& F_2 \& ... \& F_n)}$

Quantifiers

all and **exists**

if F is a formula,

$Q_1, Q_2, ...Q_n$ are quantifiers and

$x_1, x_2, ..., x_n$ are names, then

$\underline{(Q_1 x_1 Q_2 x_2 ... Q_n x_n F)}$

If F is a non-negated formula, then

$\underline{(-F)}$

Notation in OTTER

Formulas:
Always surrounded by parenthesis

A formula is converted to a set of clauses by the system

```
--------------A Formula---------------
(P | Q | R).

--------------A Clause----------------
P | Q | R.
```

OTTER formula list

If someone is alive at a certain time,
then he is not dead
$$\forall_{x,t} alive(x,t) \rightarrow -dead(x,t)$$

Nobody lives longer than 150
(If a given time is greater than 150 years after he is born, then he is dead)
$$\forall_{x,t1,t2}(born(x,t1) \wedge gt(t2-t1,150)) \rightarrow dead(x,t2)$$

Alive means not dead
$$\forall_{x,t} alive(x,t) \rightarrow \neg dead(x,t)$$

```
formula_list(usable).
(all x all t (alive(x,t) -> -dead(x,t))).
(all x all t1 all t2 ((born(x,t1) & (t2-t1 > 150))
                                  -> dead(x,t2))).
end_of_list.
```

Resolution Works on Clauses, not Formulas

All Formulas have to be converted to a set of clauses.

Eliminate implication ($\rightarrow$)

$$(A \rightarrow B) \equiv -A \mid B$$

Reduce scope of negative symbols

$$\neg(A \mid B) \equiv -A \wedge -B$$
$$\neg(A \wedge B) \equiv -A \mid -B$$

Eliminate Existential Quantifiers

Use skolem functions

Resolution Works on Clauses, not Formulas

Convert to prenex form

Convert formula to a string of quantifiers followed by a quantifier free formula, called a matrix

$\forall_x \forall_y ... P(x, y, ...)$

Convert Matrix to Conjuctive normal form

Clauses separated by AND

$X1 \wedge (X2 \mid X3) \wedge X4 \wedge (X5 \mid X6 \mid X7 \mid X8)$

Eliminate $\wedge$ symbols

Convert the clauses separated by AND to a set of clauses

$C_1 \wedge C_2 \wedge ... \wedge C_n \equiv \{C_1, C_2 ... C_n\}$

Rename Variables

Rename so no variable name appears in more than one clause (standardizing variables apart)

OTTER converts formulas to clauses

```
formula_list(usable).
all x t (alive(x,t)-> -dead(x,t)).
all x t1 t2 (born(x,t1)&t2-t1>150->dead(x,t2)).
pompeian(marcus)&loyalto(marcus,cesear).
end_of_list.

-------> usable clausifies to:

list(usable).
1 [] -alive(X1,X2)| -dead(X1,X2).
2 [] -born(X3,X4)| -(X5-X4>150)|dead(X3,X5).
3 [] pompeian(marcus).
4 [] loyalto(marcus,cesear).
end_of_list.
```

Disadvantage of Transformation

Compare

$$\forall_{x,t}(born(x,t1) \wedge (t2 - t1 > 150)) \rightarrow dead(x,t2))$$

and

$$\neg born(X3, X4) \mid \neg(X5 - X4 > 150) \mid dead(X3, X5)$$

Some of the original intuition is lost in the transformation

Loss of information

Resolution Inference Rules

$Clause_1$	$Clause_2$	Resolvent	
P	$\neg P$	empty	Contradiction
P	$\neg P \wedge Q$	Q	Modus Podens
$P \wedge Q$	$\neg P \wedge Q$	Q	Merge
$\neg P \wedge Q$	$\neg Q \wedge R$	$\neg P \wedge R$	Chaining

Note:
$\neg P \wedge Q$
is the clause form of the formula
$P \rightarrow Q$

These inference rules *reduce* the clauses to a set of *smaller*, but equivalent clauses

Proof By Contradiction

Refutation

Assume
a set of conditions

Assume
The negative of what is to be proved

Show that they are inconsistent
(i.e. a contradiction)

Empty Clause

In the set of clauses
$\{..., \neg P, P, ...\}$

Resolves to
$\{..., NIL, ...\}$

A Contradiction

What is the meaning of
$\{\neg P, P\}$?

From the transformation rules
Would come from the formula
$\neg P \wedge P$
which can never be true

To Prove, Marcus is Dead

Assumptions:

$$\forall_{x,t} alive(x,t) \rightarrow \neg dead(x,t)$$
$$\forall_{x,t1,t2}(born(x,t1) \wedge gt(t2-t1,150)) \rightarrow dead(x,t2)$$
$$born(marcus,40)$$

To Prove

$$\neg alive(marcus,1995)$$

Set up contradiction

$$alive(marcus,1995)$$

Thus need to refute:

$$\forall_{x,t} alive(x,t) \rightarrow \neg dead(x,t)$$
$$\forall_{x,t1,t2}(born(x,t1) \wedge gt(t2-t1,150)) \rightarrow dead(x,t2)$$
$$born(marcus,40)$$
$$alive(marcus,1995)$$

OTTER Resolution: Set of Clauses

```
1 [] -alive(X1,X2)| -dead(X1,X2).
2 [] -born(X3,X4)| -(X5-X4>150)|dead(X3,X5).
3 [] born(marcus,40).
4 [] alive(marcus,1994).
```

OTTER Resolution

Clause 1:
$\neg alive(X1, X2) \mid \neg dead(X1, X2).$

Clause 4
$alive(marcus, 1994).$

Unify
X1 with marcus
X2 with 1994

Resolution
$\neg alive(marcus, 1994) \mid \neg dead(marcus, 1994).$
$alive(marcus, 1994).$
to yield

Clause 5 (new)
$\neg dead(marcus, 1994).$

```
given clause #1: (wt=3) 4 [] alive(marcus,1994).
** KEPT (wt=3): 5 [binary,4.1,1.1] -dead(marcus,1994).
```

OTTER Resolution

Clause 5
$$-dead(marcus, 1994).$$

Clause 2
$$-born(X3, X4) \mid -(X5 - X4 > 150) \mid dead(X3, X5).$$

Resolution
Clause 6 (new)
$$-born(marcus, X4) \mid -(1994 - X4 > 150).$$

```
given clause #2:
(wt=3) 5 [binary,4.1,1.1] -dead(marcus,1994).
** KEPT (wt=8):
6 [binary,5.1,2.3] -born(marcus,A)| -(1994-A>150).
```

OTTER Resolution

Clause 6

$$-born(marcus, X4) \mid -(1994 - X4 > 150).$$

Clause 3

$$born(marcus, 40).$$

Resolution

$$\neg(1994 - 40 > 150).$$

FALSE

Contradiction

Proved

```
given clause #3:
(wt=8) 6 [binary,5.1,2.3] -born(marcus,A)| -(1994-A>150).
** KEPT (wt=0):
7 [binary,6.1,3.1,demod] $F.

-----> EMPTY CLAUSE at   0.39 sec ---->
7 [binary,6.1,3.1,demod] $F.
```

OTTER: The complete proof

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

1 [] -alive(X1,X2)| -dead(X1,X2).
2 [] -born(X3,X4)| -(X5-X4>150)|dead(X3,X5).
3 [] born(marcus,40).
4 [] alive(marcus,1994).
5 [binary,4.1,1.1] -dead(marcus,1994).
6 [binary,5.1,2.3] -born(marcus,A)| -(1994-A>150).
7 [binary,6.1,3.1,demod] $F.

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

Set Of Support Strategy

Strategy
Directs the program as to which clause is to be worked on

Set of Support

By Input
Clauses that have been put into this set by input

Inference Rule
Clauses that have been obtained by an inference rule

The inference rules use one rule from the
Set of Support Clauses

OTTER: Lists of formulas, clauses and Set of Support

```
formula_list(usable).
(all x all t (alive(x,t) -> -dead(x,t))).
(all x all t1 all t2 ((born(x,t1) & (t2-t1 > 150))
                           -> dead(x,t2))).
end_of_list.

list(usable).
born(marcus,40).
end_of_list.

list(sos).
alive(marcus,1994).
end_of_list.
```

Set of Support

```
*********** Set of Support *************
4 [] alive(marcus,1994).
********* Usable Clauses ***************
1 [] -alive(X1,X2)| -dead(X1,X2).
2 [] -born(X3,X4)| -(X5-X4>150)|dead(X3,X5).
3 [] born(marcus,40).
---------------------------------------------------------------
given clause #1: (wt=3) 4 [] alive(marcus,1994).

  0 [binary,4.1,1.1] -dead(marcus,1994).
  after demodulation: 0 [binary,4.1,1.1] -dead(marcus,1994).
** KEPT (wt=3): 5 [binary,4.1,1.1] -dead(marcus,1994).
---------------------------------------------------------------
*********** New Set of Support *************
5 [binary,4.1,1.1] -dead(marcus,1994).
********* New Usable Clauses ***************
1 [] -alive(X1,X2)| -dead(X1,X2).
2 [] -born(X3,X4)| -(X5-X4>150)|dead(X3,X5).
3 [] born(marcus,40).
4 [] alive(marcus,1994).
```

```
*********** Set of Support *************
5 [binary,4.1,1.1] -dead(marcus,1994).
********* Usable Clauses ***************
1 [] -alive(X1,X2)| -dead(X1,X2).
2 [] -born(X3,X4)| -(X5-X4>150)|dead(X3,X5).
3 [] born(marcus,40).
4 [] alive(marcus,1994).
--------------------------------------------------------------
given clause #2: (wt=3) 5 [binary,4.1,1.1] -dead(marcus,1994).

  0 [binary,5.1,2.3] -born(marcus,A)| -(1994-A>150).
  after demodulation: 0
         [binary,5.1,2.3] -born(marcus,A)| -(1994-A>150).
** KEPT (wt=8): 6 [binary,5.1,2.3] -born(marcus,A)| -(1994-A>150).
--------------------------------------------------------------
*********** New Set of Support *************
6 [binary,5.1,2.3] -born(marcus,A)| -(1994-A>150).
********* New Usable Clauses **************
1 [] -alive(X1,X2)| -dead(X1,X2).
2 [] -born(X3,X4)| -(X5-X4>150)|dead(X3,X5).
3 [] born(marcus,40).
4 [] alive(marcus,1994).
5 [binary,4.1,1.1] -dead(marcus,1994).
```

```
*********** Set of Support *************
6 [binary,5.1,2.3] -born(marcus,A)| -(1994-A>150).
********* Usable Clauses ***************
1 [] -alive(X1,X2)| -dead(X1,X2).
2 [] -born(X3,X4)| -(X5-X4>150)|dead(X3,X5).
3 [] born(marcus,40).
4 [] alive(marcus,1994).
5 [binary,4.1,1.1] -dead(marcus,1994).

given clause #3:
   (wt=8) 6 [binary,5.1,2.3] -born(marcus,A)| -(1994-A>150).

  0 [binary,6.1,3.1] -(1994-40>150).
  after demodulation: 0 [binary,6.1,3.1,demod] -$T.
** KEPT (wt=0): 7 [binary,6.1,3.1,demod] $F.
```

A simplification of Notation in OTTER

```
make_evaluable(_-_,$DIFF(_,_)).
make_evaluable(_+_,$SUM(_,_)).
make_evaluable(_<_,$LT(_,_)).
make_evaluable(_>_,$GT(_,_)).
```

Complete Program

```
% Set Binary Resolution
set(binary_res).
% Capitals are variables and small are constants
set(prolog_style_variables).
% Turn on all the print flags
set(very_verbose).

% The list of operators (rewritten to be more readable)
make_evaluable(_-_, $DIFF(_,_)).
make_evaluable(_>_, $GT(_,_)).

% Set of Usable Formulas (will be converted to clauses)
formula_list(usable).
(all x all t (alive(x,t) -> -dead(x,t))).
(all x all t1 all t2 ((born(x,t1) & (t2-t1 > 150))
                              -> dead(x,t2))).
end_of_list.

% Set of Usable Clauses
list(usable).
born(marcus,40).
end_of_list.

% Set of Support
list(sos).
alive(marcus,1994).
```

```
end_of_list.
```

UR-Resolution

One clause is a non-unit clause

The rest are unit clauses
(i.e. there can be more than one)

The result is a unit clause

Binary resolution involves only two clauses
UR-Resolution can involve several

Example of UR-Resolution

```
1 [] -alive(X1,X2)| -dead(X1,X2).
2 [] -born(X3,X4)| -(X5-X4>150)|dead(X3,X5).
3 [] pompeian(marcus).
4 [] loyalto(marcus,cesear).
5 [] born(marcus,40).

list(sos).
6 [] alive(marcus,1994).
end_of_list.

=========== start of search ===========

given clause #1: (wt=3) 6 [] alive(marcus,1994).

  0 [ur,6,1] -dead(marcus,1994).
  after demodulation: 0 [ur,6,1] -dead(marcus,1994).
** KEPT (wt=3): 7 [ur,6,1] -dead(marcus,1994).

given clause #2: (wt=3) 7 [ur,6,1] -dead(marcus,1994).

  0 [ur,7,2,5] -(1994-40>150).
  after demodulation: 0 [ur,7,2,5,demod] -$T.
** KEPT (wt=0): 8 [ur,7,2,5,demod] $F.

-----> EMPTY CLAUSE at   0.07 sec ----> 8 [ur,7,2,5,demod] $F.
```

Hyperresolution

One clause must be a *Negative* or a *Mixed* Clause
(nucleus)

The remaining clauses must be *Positive*
(satellites)

The number of positive clauses
must equal
the number of negative literals

The result is a postive clause

The negative literals
and their corresponding positive literals
are eliminated
(the remaining literals are ORed together)

Example where Hyperresolution does not work

```
1 [] -alive(X1,X2)| -dead(X1,X2).
2 [] -born(X3,X4)| -(X5-X4>150)|dead(X3,X5).
3 [] pompeian(marcus).
4 [] loyalto(marcus,cesear).
5 [] born(marcus,40).

list(sos).
6 [] alive(marcus,1994).
end_of_list.
=========== start of search ===========

given clause #1: (wt=3) 6 [] alive(marcus,1994).

Search stopped because sos empty.ϒ

Search stopped because sos empty.
```

Use of Hyperresolution

Typical Use

Formula

$$((P_1 \wedge P_2 \wedge ... \wedge P_n) \rightarrow Q)$$

As Clause

$$\neg P_1 \mid \neg P_2 \mid ... \mid \neg P_n \mid Q$$

Instead of n binary resolutions:
in one step,
the negative literals are removed

Cannibal and Missionary Problem

A State Space Represention

A Given State AND a set of conditions
implies
A new state

$$
\begin{gathered}
state(MBS, CSB, BP)\wedge \\
pick(M) \wedge pick(C)\wedge \\
(M \leq MBS) \wedge (C \leq CBS)\wedge \\
(M + C > 0) \wedge (M + C \leq 2)\wedge \\
((C >= M) \mid (C = 0))\wedge \\
((CBS - C \geq MBS - M) \mid (CBS - C = 0))\wedge \\
(((3 - CBS) + C \geq (3 - MBS) + M) \mid ((3 - CBS) + C == 0))\wedge \\
\rightarrow \\
state((3 - MBS) + M, (3 - CBS) + C, otherside(BP)).
\end{gathered}
$$

$$
\begin{gathered}
pick(0). \\
pick(1). \\
pick(2).
\end{gathered}
$$

state(3,3,west)

With Binary Resolution: First Few Steps

```
1 [] -state(MBS,CBS,BP)| -pick(M)| -pick(C)| -(M<=MBS)| -(C<=CBS)| -(M+C>0)| -(M+C<=2)| -(C>=M||C==0)| -(CBS-C>=MBS-M||CBS-C==0)| -((3-CBS)+C>= (3-MBS)+M|| (3-CBS)+C==0)
                     |state((3-MBS)+M,(3-CBS)+C,otherside(BP)).
2 [] pick(0).
3 [] pick(1).
4 [] pick(2).
5 [] -state(3,3,east).
6 [] state(3,3,west).
7 [] otherside(west)=east.
8 [] otherside(east)=west.
9 [binary,6.1,1.1,demod,7] -pick(A)| -pick(B)| -(A<=3)| -(B<=3)| -(A+B>0)| -(A+B<=2)| -(B>=A||B==0)|
             -(3-B>=3-A||3-B==0)| -(0+B>=0+A||0+B==0)
             |state(0+A,0+B,east).
10 [factor,9,1,2,factor_simp] -pick(A)| -(A<=3)| -(A+A>0)| -(A+A<=2)| -(A>=A||A==0)|
             -(3-A>=3-A||3-A==0)| -(0+A>=0+A||0+A==0)
             |state(0+A,0+A,east).
11 [binary,10.1,3.1,demod] state(1,1,east).
12 [binary,10.8,1.1,demod,8] -pick(A)| -(A<=3)| -(A+A>0)| -(A+A<=2)| -(A>=A||A==0)|
             -(3-A>=3-A||3-A==0)| -(0+A>=0+A||0+A==0)| -pick(B)| -pick(C)| -(B<=0+A)|
             -(C<=0+A)| -(B+C>0)| -(B+C<=2)| -(C>=B||C==0)|-((0+A)-C>= (0+A)-B|| (0+A)-C==0)|
             -((3- (0+A))+C>= (3- (0+A))+B|| (3- (0+A))+C==0)
             |state((3- (0+A))+B,(3- (0+A))+C,west).
13 [factor,12,1,8] -pick(A)| -(A<=3)| -(A+A>0)| -(A+A<=2)| -(A>=A||A==0)| -(3-A>=3-A||3-A==0)|
             -(0+A>=0+A||0+A==0)| -pick(B)| -(A<=0+A)| -(B<=0+A)| -(A+B>0)| -(A+B<=2)|
             -(B>=A||B==0)| -((0+A)-B>= (0+A)-A|| (0+A)-B==0)|
             -((3- (0+A))+B>= (3- (0+A))+A|| (3- (0+A))+B==0)
             |state((3- (0+A))+A,(3- (0+A))+B,west).
19 [binary,11.1,1.1,demod,8] -pick(A)| -pick(B)| -(A<=1)| -(B<=1)| -(A+B>0)| -(A+B<=2)|
             -(B>=A||B==0)| -(1-B>=1-A||1-B==0)| -(2+B>=2+A||2+B==0)
             |state(2+A,2+B,west).
120 [binary,19.2,3.1,demod] -pick(A)| -(A<=1)| -(A+1>0)| -(A+1<=2)| -(1>=A||$F)| -(0>=1-A||$T)| -(3>=2+A||$F)
             |state(2+A,3,west).
138 [binary,120.1,2.1,demod] state(2,3,west).
```

With Hyper-Resolution: First Few Steps

```
given clause #1: (wt=4) 6 [] state(3,3,west).
** KEPT (wt=4): 9 [hyper,6,1,4,2,eval,eval,eval,eval,eval,eval,eval,demod,7]
                         state(2,0,east).
** KEPT (wt=4): 10 [hyper,6,1,3,3,eval,eval,eval,eval,eval,eval,eval,demod,7]
                         state(1,1,east).
** KEPT (wt=4): 11 [hyper,6,1,3,2,eval,eval,eval,eval,eval,eval,eval,demod,7]
                          state(1,0,east).

given clause #2: (wt=4) 9 [hyper,6,1,4,2,eval,eval,eval,eval,eval,eval,eval,demod,7] state(2,0,east).
** KEPT (wt=4): 12 [hyper,9,1,3,2,eval,eval,eval,eval,eval,eval,eval,demod,8]
                          state(2,3,west).

given clause #3: (wt=4) 10 [hyper,6,1,3,3,eval,eval,eval,eval,eval,eval,eval,demod,7]
                           state(1,1,east).
```

Paramodulation

An Equality Substitution

The clause containing the equality:
The *FROM* clause

The clause into which the substitution is made:
The *INTO* clause

After Unification of
one side of the FROM clause
with a term in the INTO clause
The otherside of the FROM is substituted in

Generalization of Equality

Paramodulation Example

$$sum(0, x) = x$$

$$sum(y, minus(y)) = 0$$

Unification
$sum(0, x)$ and $sum(y, minus(y))$
y is 0
x is minus(y) which is minus(0)

Result
FROM $sum(0, minus(0)) = minus(0)$
TO $sum(0, minus(0)) = 0$
$minus(0) = 0$

Note that is not simple syntactic substitution.
Unification played a major role

OTTER: Paramodulation

```
set(para_from).

1 [] sum(x,minus(x))=0.
2 [] minus(0)!=0.
3 [] sum(0,y)=y.

=========== start of search ===========
given clause #1: (wt=5) 3 [] sum(0,y)=y.

  0 [para_from,3.1.1,3.1.1] x=x.
** KEPT (wt=3): 4 [para_from,3.1.1,3.1.1] x=x.

  0 [para_from,3.1.1,1.1.1] minus(0)=0.
** KEPT (wt=4): 5 [para_from,3.1.1,1.1.1] minus(0)=0.

----> UNIT CONFLICT at   0.03 sec ---->
                  6 [binary,5.1,2.1] $F.
```

Demodulation

Very Similar to Paramodulation

Allows only variable replacement in equality
not in the term to be substituted
$brother(father(x) = uncle(x)$
$age(brother(father(john)), 55)$
$age(uncle(john))$

Demodulation discards the original version
Paramodulation keeps both

Demodulation is tried repeatedly upon success
Paramodulation is tried only once

Subsumption

Discards a clause that is
a duplicate
or
a less general clause

$older(father(x), x)$
subsumes
$older(father(Ann), Ann)$

Forward Subsumption
When a newly generated clause is discarded
because a previously retained clause subsumes it

Back Subsumption
When a newly generated clause is used to discard
a previously retained clauses

Complex Demodulators as programming language

```
list(demodulators).
%%%%% factorial for nonnegative integers
factorial(N) =
   $IF(N==0,
       1,
       N*factorial(N-1)).

%%%%% gcd for nonnegative integers
gcd(X,Y) =
   $IF(X==0,
       Y,
       $IF(Y==0,
           X,
           $IF(X<Y,
               gcd(X,Y-X),
               gcd(Y,X-Y)))).
%%%%% member of list function
member(X,[]) = $F.
member(X,[H|T]) = $IF($ID(X,H),
                      $T,
                      member(X,T)).
end_of_list.
```