Computational Logic Lecture 9

Resolution Preliminaries

By Prof. Michael Genesereth, Computer Science, Stanford Univ

Spring 2005

Modified by Charles Ling. Use with permission

Resolution Principle

The *Resolution Principle* is a rule of inference.

Using the Resolution Principle alone (without axiom schemata or other rules of inference), it is possible to build a theorem prover that is sound and complete for all of Relational Logic.

The search space using the Resolution Principle is much smaller than with standard axiom schemata.

Plan

First Lecture:

- Unification
- Relational Clausal Form

Second Lecture:

- Resolution Principle
- Resolution Theorem Proving

Third Lecture:

- True or False Questions
- Fill in the Blank Questions
- Residue

Fourth Lecture:

- Strategies to enhance efficiency

Clausal Form

Relational resolution works only on expressions in *clausal form*.

Fortunately, it is possible to convert any set of Relational Logic sentences into an equally satisfiable set of sentences in clausal form.

Clausal Form

A *literal* is either an atomic sentence or a negation of an atomic sentence.
A *clausal sentence* is either a literal or a disjunction of literals.
A *clause* is a set of literals (interpreted as disjunction).

$\{p(a)\}$ [Pa in LogiCola]

$\{\neg p(a)\}$

$\{p(a), q(b)\}$

The empty clause { } is unsatisfiable
Convert to clauses: leave only universal variables!

INSEADO

Implications Out:

$$\varphi_1 \Rightarrow \varphi_2 \quad \rightarrow \quad \neg\varphi_1 \vee \varphi_2$$
$$\varphi_1 \Leftarrow \varphi_2 \quad \rightarrow \quad \varphi_1 \vee \neg\varphi_2$$
$$\varphi_1 \Leftrightarrow \varphi_2 \quad \rightarrow \quad (\neg\varphi_1 \vee \varphi_2) \wedge (\varphi_1 \vee \neg\varphi_2)$$

Negations In:

$$\neg\neg\varphi \quad \rightarrow \quad \varphi$$
$$\neg(\varphi_1 \wedge \varphi_2) \quad \rightarrow \quad \neg\varphi_1 \vee \neg\varphi_2$$
$$\neg(\varphi_1 \vee \varphi_2) \quad \rightarrow \quad \neg\varphi_1 \wedge \neg\varphi_2$$
$$\neg\forall \nu.\varphi \quad \rightarrow \quad \exists \nu.\neg\varphi$$
$$\neg\exists \nu.\varphi \quad \rightarrow \quad \forall \nu.\neg\varphi$$

Inseado (continued)

Standardize variables (also between different clauses)

$$\forall x.p(x) \vee \forall x.q(x) \quad \rightarrow \quad \forall x.p(x) \vee \forall y.q(y)$$

Existentials Out (Outside in): Introducing Skolem constants and functions.

$$\exists x.p(x) \quad \rightarrow \quad p(a)$$

$$\forall xy.(p(x) \wedge \exists z.q(x,y,z)) \quad \rightarrow \quad \forall xy.(p(x) \wedge q(x,y,f(x,y)))$$

(If there are universal quantifiers outside, use a function once for all. In LogiCola, we may drop initial universal first, then drop existential with a new constant (say e), but then re-use the universal with the new constant e – so these two approaches are essentially the same)

Inseado (continued)

Alls Out (**all variables must be universal**!)

$$\forall xy.(p(x) \wedge q(x, y, f(x, y))) \quad \rightarrow \quad p(x) \wedge q(x, y, f(x, y))$$

Distribution

$$\varphi_1 \vee (\varphi_2 \wedge \varphi_3) \quad \rightarrow \quad (\varphi_1 \vee \varphi_2) \wedge (\varphi_1 \vee \varphi_3)$$

$$(\varphi_1 \wedge \varphi_2) \vee \varphi_3 \quad \rightarrow \quad (\varphi_1 \vee \varphi_3) \wedge (\varphi_2 \vee \varphi_3)$$

Inseado (continued)

Operators Out

$$\varphi_1 \wedge \ldots \wedge \varphi_n \rightarrow \varphi_1$$

$$\ldots$$

$$\varphi_n$$

$$\varphi_1 \vee \ldots \vee \varphi_n \rightarrow \{\varphi_1, \ldots, \varphi_n\}$$

Example

$\exists y.(g(y) \wedge \forall z.(r(z) \Rightarrow f(y,z)))$

I $\exists y.(g(y) \wedge \forall z.(\neg r(z) \vee f(y,z)))$

N $\exists y.(g(y) \wedge \forall z.(\neg r(z) \vee f(y,z)))$

S $\exists y.(g(y) \wedge \forall z.(\neg r(z) \vee f(y,z)))$

E $g(greg) \wedge \forall z.(\neg r(z) \vee f(greg,z)))$

A $g(greg) \wedge (\neg r(z) \vee f(greg,z)))$

D $g(greg) \wedge (\neg r(z) \vee f(greg,z)))$

O $\{g(greg)\}$

$\{\neg r(z), f(greg,z)\}$

Example (negated… for conclusion)

$$\neg\exists y.(g(y) \wedge \forall z.(r(z) \Rightarrow f(y,z)))$$

I $\quad \neg\exists y.(g(y) \wedge \forall z.(\neg r(z) \vee f(y,z)))$

N $\quad \neg\exists y.(g(y) \wedge \forall z.(\neg r(z) \vee f(y,z)))$

$$\forall y.\neg(g(y) \wedge \forall z.(\neg r(z) \vee f(y,z)))$$

$$\forall y.(\neg g(y) \vee \neg\forall z.(\neg r(z) \vee f(y,z)))$$

$$\forall y.(\neg g(y) \vee \exists z.\neg(\neg r(z) \vee f(y,z)))$$

$$\forall y.(\neg g(y) \vee \exists z.(\neg\neg r(z) \wedge \neg f(y,z)))$$

$$\forall y.(\neg g(y) \vee \exists z.(r(z) \wedge \neg f(y,z)))$$

S $\quad \forall y.(\neg g(y) \vee \exists z.(r(z) \wedge \neg f(y,z)))$

Example (continued)

$\forall y.(\neg g(y) \vee \exists z.(r(z) \wedge \neg f(y,z)))$

E $\forall y.(\neg g(y) \vee (r(h(y)) \wedge \neg f(y,h(y))))$

A $\neg g(y) \vee (r(h(y)) \wedge \neg f(y,h(y)))$

D $(\neg g(y) \vee r(h(y))) \wedge (\neg g(y) \vee \neg f(y,h(y)))$

O $\{\neg g(y) \vee r(h(y))\}$

$\{\neg g(y) \vee \neg f(y,h(y))\}$

6. Romeo loves all females.

No females love Romeo.

Juliet is female.

∴ Romeo loves someone who doesn't love him. [Use Lxy, r, Fx, and j.]

```
  1     (x)(Fx ⊃ Lrx)
* 2     ~(∃x)(Fx · Lxr)
  3     Fj
    [∴ (∃x)(Lrx · ~Lxr)
* 4   ┌ asm: ~(∃x)(Lrx · ~Lxr)
  5   │ ∴ (x)~(Fx · Lxr)    {from 2}
  6   │ ∴ (x)~(Lrx · ~Lxr)  {from 4}
* 7   │ ∴ (Fj ⊃ Lrj)        {from 1}
  8   │ ∴ Lrj               {from 3 and 7}
* 9   │ ∴ ~(Fj · Ljr)       {from 5}
 10   │ ∴ ~Ljr              {from 3 and 9}
*11   │ ∴ ~(Lrj · ~Ljr)     {from 6}
 12   └ ∴ Ljr               {from 8 and 11}
 13   ∴ (∃x)(Lrx · ~Lxr)    {from 4; 10
        contradicts 12}
```

Converting to clauses and use Resolution:

1~F(x) v L(r, x): clause

2~F(x) v ~L(x, r): clause

3F(j): clause

4Neg goal: ~L(r, x) v L(x, r): clause

5L(r, j) : 1 and 3, x matches j

6L(j, r) : 5 and 4, x matches j

7~F(j) : 2 and 6, x matches j

8{} : 3 and 7

7. In all cases, if a first thing caused a second then the first exists before the second.
Nothing exists before it exists.
∴ Nothing caused itself. [Use Cxy and Bxy (for "x exists before y exists").]

```
  1    (x)(y)(Cxy ⊃ Bxy)
* 2    ~(∃x)Bxx
   [∴ ~(∃x)Cxx
* 3  ┌ asm: (∃x)Cxx
  4  │ ∴ (x)~Bxx    {from 2}
  5  │ ∴ Caa    {from 3}
  6  │ ∴ (y)(Cay ⊃ Bay)    {from 1}
  7  │ ∴ ~Baa    {from 4}
* 8  │ ∴ (Caa ⊃ Baa)    {from 6}
  9  └ ∴ Baa    {from 5 and 8}
 10  ∴ ~(∃x)Cxx    {from 3; 7 contradicts 9}
```

Use Resolution:

1 ~C(x, y) v B(x, y): clause

2 ~B(x, x)

3 neg goal: C(a, a)

4 B(a, a) :1 and 3, x=a, y=a

5 { } : 2 and 4, x=a

Example

Everybody loves somebody. Everybody loves a lover. Show that everybody loves everybody.

$\forall x.\exists y.loves(x,y)$

$\forall u.\forall v.\forall w.(loves(v,w) \Rightarrow loves(u,v))$

$\neg\forall x.\forall y.loves(x,y)$

$\{loves(x, f(x))\}$

$\{\neg loves(v,w), loves(u,v)\}$

$\{\neg loves(jack, jill)\}$

Use different variable names!

Try S/I rules:

4~loves(a, b) : from 3
5Exist y loves(a, y) from 3
6loves(a, c) from 5
7loves(b, d) similar to 5,6
8loves(b, d) > loves(a, b): from 2
9loves(a, b) : from 6, 7, I rule
10RAA : 4 and 8

(Note: can have many forms of 7)

Try resolution:

4~loves(jill, w): 2 and 3
5{}, 4 and 1, x=jill, w=f(jill)

(So "matching" can be more complicated)

Clausal Form

Bad News: The result of converting a set of sentences is not necessarily logically equivalent to the original set of sentences. Why? Introduction of Skolem constants and functions.

Good News: The result of converting a set of sentences is satisfiable if and only if the original set of sentences is satisfiable. Important because we use satisfiability to determine logical entailment.

Difficulty with Universal Instantiation

Simple cases are easy!

$$\forall x.p(x,b)$$
$$\forall y.\neg p(a,y)$$

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

Substitutions

A *substitution* is a finite set of pairs of variables and terms, called *replacements*.

$$\{X \leftarrow a,\ Y \leftarrow f(b),\ V \leftarrow W\}$$

The result of applying a substitution σ to an expression ϕ is the expression $\phi\sigma$ obtained from ϕ by replacing every occurrence of every variable in the substitution by its replacement.

$$p(X,X,Y,Z)\{X \leftarrow a, Y \leftarrow f(b), V \leftarrow W\} = p(a,a,f(b),Z)$$

Cascaded Substitutions

$$r\{x,y,z\}\{x\leftarrow a,\ y\leftarrow f(u),\ z\leftarrow v\}=r\{a,f(u),v\}$$

$$r\{a,f(u),v\}\{u\leftarrow d,\ v\leftarrow e\}=r(a,f(d),e)$$

$$r\{x,y,z\}\{x\leftarrow a,y\leftarrow f(d),z\leftarrow e\}=r(a,f(d),e)$$

Composition of Substitutions

The *composition* of substitution σ and τ is the substitution (written *compose(*σ,τ*)* or, more simply, στ) obtained by
(1) applying τ to the replacements in σ
(2) adding to σ pairs from τ with different variables
(3) deleting any assignments of a variable to itself.

$$\{x \leftarrow a,\ y \leftarrow f(u),\ z \leftarrow v\}\{u \leftarrow d, v \leftarrow e, z \leftarrow g\}$$

$$= \{x \leftarrow \mathrm{a}, y \leftarrow f(d), z \leftarrow e\}\{u \leftarrow d, v \leftarrow e, z \leftarrow g\}$$

$$= \{x \leftarrow \mathrm{a}, y \leftarrow f(d), z \leftarrow e, u \leftarrow d, v \leftarrow e\}$$

Unification

A substitution σ is a *unifier* for an expression ϕ and an expression ψ if and only if $\phi\sigma=\psi\sigma$.

`p(X,Y){X←a,Y←b,V←b}=p(a,b)`
`p(a,V){X←a,Y←b,V←b}=p(a,b)`

If two expressions have a unifier, they are said to be *unifiable*. Otherwise, they are *nonunifiable*.

`p(X,X)`
`p(a,b)`

Non-Uniqueness of Unification

Unifier 1:

$p(X,Y)\{X \leftarrow a, Y \leftarrow b, V \leftarrow b\} = p(a,b)$

$p(a,V)\{X \leftarrow a, Y \leftarrow b, V \leftarrow b\} = p(a,b)$

Unifier 2:

$p(X,Y)\{X \leftarrow a, Y \leftarrow f(W), V \leftarrow f(W)\} = p(a,f(W))$

$p(a,V)\{X \leftarrow a, Y \leftarrow f(W), V \leftarrow f(W)\} = p(a,f(W))$

Unifier 3:

$p(X,Y)\{X \leftarrow a, Y \leftarrow V\} = p(a,V)$

$p(a,V)\{X \leftarrow a, Y \leftarrow V\} = p(a,V)$

Most General Unifier

A substitution σ is a *most general unifier* (*mgu*) of two expressions if and only if it is as general as or more general than any other unifier.

Theorem: If two expressions are unifiable, then they have an mgu that is unique up to variable permutation.

```
p(X,Y){X←a,Y←V}=p(a,V)
p(a,V){X←a,Y←V}=p(a,V)

p(X,Y){X←a,V←Y}=p(a,Y)
p(a,V){X←a,V←Y}=p(a,Y)
```

[MGU: "commit the least principle" ☺
May omit the next few slides on how to find mgu.
Usually easy to in simple proofs]

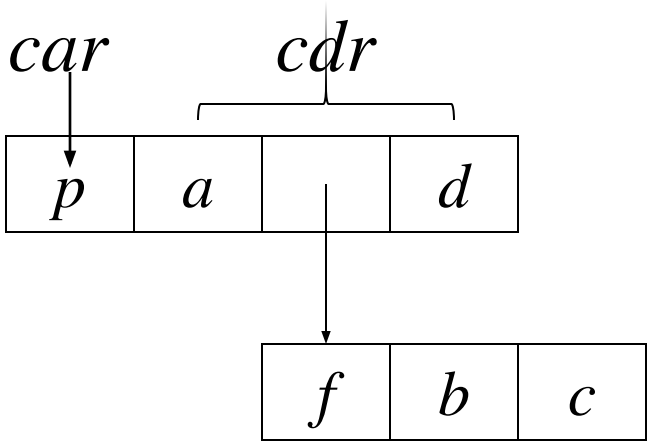
Expression Structure

Each expression is treated as a sequence of its immediate subexpressions.

Linear Version:

$$p(a, f(b, c), d)$$

Structured Version:



Most General Unification

```
function mgu (x, y, s)
  {if x = y then s
    else if varp(x) then mguvar(x, y, s)
    else if atom(x) then {if  varp(y) then mguvar(y, x, s)
                          else if atom(y) then if x = y then s}
    else if varp(y) then mguvar(y, x, s)
    else if atom(y) then false
    else if s←mgu(car(x),car(y), s) then mgu(cdr(x), cdr(y), s)}

function mguvar (x, y, s)
  {var dum;
    if dum ← assoc(x, s) then mgu(right(dum), y, s)
    else compose(s,{x ← plug(y,s)})}
```

Example

Call: *mgu(*`p(X,b), p(a,Y), {}`*)*

Call: *mgu(*`p, p, {}`*)*
Exit: `{}`

Call: *mgu(*`X, a, {}`*)*
Exit: `{X←a}`

Call: *mgu(*`b, Y, {X←a}`*)*
Exit: `{X←a, Y←b}`

Exit: `{X←a, Y←b}`

Example

Call: *mgu(*`p(X,X)`, `p(a,b)`, { }*)*

Call: *mgu(*`p`, `p`, { }*)*
Exit: { }

Call: *mgu(*`X`, `a`, { }*)*
Exit: {`X`←`a`}

Call: *mgu(*`X`, `b`, {`X`←`a`}*)*
Call: *mgu(*`a`, `b`, {`X`←`a`}*)*
Exit: `false`
Exit: `false`

Exit: `false`

Example

Call: *mgu(*`p(f(X),f(X)), p(Y,f(a)), {}`*)*
  Call: *mgu(*`p, p, {}`*)*
  Exit: `{}`

  Call: *mgu(*`f(X), Y, {}`*)*
  Exit: `{Y←f(X)}`

  Call: *mgu(*`f(X), f(a), {Y←f(X)}`*)*
    Call: *mgu(*`f, f, {Y←f(X)}`*)*
    Exit: `{Y←f(X)}`
    Call: *mgu(*`X, a, {Y←f(X)}`*)*
    Exit: `{Y←f(a),X←a}`
  Exit: `{Y←f(a),X←a}`

Exit: `{Y←f(a),X←a}`

Example

Call: *mgu(*`p(X,X)`, `p(Y,f(Y))`, { }*)*
  Call: *mgu(*`p`, `p`, { }*)*
  Exit: { }

  Call: *mgu(*`X`, `Y`, { }*)*
  Exit: {`X←Y`}

  Call: *mgu(*`X`, `f(Y)`, {`X←Y`}*)*
    Call: *mgu(*`Y`, `f(Y)`, {`X←Y`}*)*
    Exit: {`X←f(Y)`, `Y←f(Y)`}
  Exit: {`X←f(Y)`, `Y←f(Y)`}
Exit: {`X←f(Y)`, `Y←f(Y)`}

Problem

Circularity Problem:

$$\{X \leftarrow f(Y), Y \leftarrow f(Y)\}$$

Unification Problem:

$$p(X,X)\{X \leftarrow f(Y), Y \leftarrow f(Y)\} = p(f(Y), f(Y))$$

$$p(Y,f(Y))\{X \leftarrow f(Y), Y \leftarrow f(Y)\} = p(f(Y), f(f(Y)))$$

Semantic Problem:

```
~hates(X,X)
 hates(Y,f(Y))
```

Solution

Before assigning a variable to an expression, first check that the variable does not occur within that expression.

This is called, oddly enough, the *occur check* test.

Prolog does not do the occur check (and is proud of it).

Most General Unification (revised)

function *mguvar (x, y, s)*
{**var** *dum*;
if *dum* ← *assoc(x, s)* **then** *mgu(right(dum), y, s)*
else if *mguchkp(x, y, s)* **then** *false*
else *compose(s,{x ← plug(y,s)})*}

function *mguchkp (p, q, s)*
{**if** *p=q* **then** *true*
else if *varp(p)* **then** *mguchkp(p, right(assoc(q, s)), s)*
else if *atom(q)* **then** *false*
else *some(lambda(x).mguchkp(p,x,s),q)*}

Example

Call: *mgu(*`p(X,X)`, `p(Y,f(Y))`, { }*)*

Call: *mgu(*`p`, `p`, { }*)*
Exit: { }

Call: *mgu(*`X`, `Y`, { }*)*
Exit: {`X`←`Y`}

Call: *mgu(*`X`, `f(Y)`, {`X`←`Y`}*)*
Call: *mgu(*`Y`, `f(Y)`, {`X`←`Y`}*)*
Exit: `false`
Exit: `false`

Exit: `false`

Computational Logic Lecture 10

Resolution Theorem Proving

Michael Genesereth Spring 2005

Plan

First Lecture:

- Unification
- Relational Clausal Form

Second Lecture:

- Resolution Principle
- Resolution Theorem Proving

Third Lecture:

- True or False Questions
- Fill in the Blank Questions
- Residue

Fourth Lecture:

- Strategies to enhance efficiency

Propositional Resolution

$$\frac{\{\varphi_1,\ldots,\varphi,\ldots,\varphi_m\} \quad \{\psi_1,\ldots,\neg\varphi,\ldots,\psi_n\}}{\{\varphi_1,\ldots,\varphi_m,\psi_1,\ldots,\psi_n\}}$$

Relational Resolution I

We have seen many such examples
But here is more formal description

$$\frac{\{\varphi_1, \ldots, \varphi, \ldots, \varphi_m\} \quad \{\psi_1, \ldots, \neg\psi, \ldots, \psi_n\}}{\{\varphi_1, \ldots, \varphi_m, \psi_1, \ldots, \psi_n\}\sigma}$$

where $\sigma = mgu(\varphi, \psi)$

Example

$\{\ p(a,y), r(y)\}$

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

$\{r(y)\}\{x \leftarrow a, y \leftarrow b\}$

$\{r(b)\}$

Example

$\{\ p(a,y),\quad r(y)\}$

$\{\neg p(x,f(x)),q(g(x))\}$

$\{r(y),q(g(x))\}\{x \leftarrow a, y \leftarrow f(a)\}$

$\{r(f(a)),q(g(a))\}$

Example

Everybody loves somebody. Everybody loves a lover. Show that everybody loves everybody.

$$\forall x.\exists y.loves(x,y)$$

$$\forall u.\forall v.\forall w.(loves(v,w) \Rightarrow loves(u,v))$$

$$\neg\forall x.\forall y.loves(x,y)$$

Example (continued)

$$\forall x.\exists y.loves(x,y)$$

$$\forall u.\forall v.\forall w.(loves(v,w) \Rightarrow loves(u,v))$$

$$\neg\forall x.\forall y.loves(x,y)$$

$$\{loves(x, f(x))\}$$

$$\{\neg loves(v,w), loves(u,v)\}$$

$$\{\neg loves(jack, jill)\}$$

Example (concluded)

1.	$\{loves(x, f(x))\}$	Premis
2.	$\{\neg loves(v, w), loves(u, v)\}$	Premis
3.	$\{\neg loves(jack, jill)\}$	Goal
4.	$\{loves(u, x)\}$	1,2
5.	$\{\}$	4,3

(Early on, we proved it in a different way)

Harry and Ralph

Every horse can outrun every dog. Some greyhound can outrun every rabbit. Show that every horse can outrun every rabbit.

$$\forall x.\forall y.(h(x) \wedge d(y) \Rightarrow f(x,y))$$

$$\exists y.(g(y) \wedge \forall z.(r(z) \Rightarrow f(y,z)))$$

$$\forall y.(g(y) \Rightarrow d(y))$$

$$\forall x.\forall y.\forall z.(f(x,y) \wedge f(y,z) \Rightarrow f(x,z))$$

$$\neg\forall x.\forall y.(h(x) \wedge r(y) \Rightarrow f(x,y))$$

(3 and 4 are extra "background" common-sense knowledge!)
Try to prove this with S/I rules by yourself!

Harry and Ralph (continued)

$$\forall x.\forall y.(h(x) \wedge d(y) \Rightarrow f(x,y))$$

$$\exists y.(g(y) \wedge \forall z.(r(z) \Rightarrow f(y,z)))$$

$$\forall y.(g(y) \Rightarrow d(y))$$

$$\forall x.\forall y.\forall z.(f(x,y) \wedge f(y,z) \Rightarrow f(x,z))$$

$$\{\neg h(x), \neg d(y), f(x,y)\}$$

$$\{g(greg)\}$$

$$\{\neg r(z), f(greg, z)\}$$

$$\{\neg g(y), d(y)\}$$

$$\{\neg f(x,y), \neg f(y,z), f(x,z)\}$$

Harry and Ralph (continued)

$$\neg \forall x. \forall y. (h(x) \wedge r(y) \Rightarrow f(x,y))$$

$\{h(harry)\}$

$\{r(ralph)\}$

$\{\neg f(harry, ralph)\}$

Harry and Ralph (concluded)

1. $\{\neg h(x), \neg d(y), f(x,y)\}$
2. $\{g(greg)\}$
3. $\{\neg r(z), f(greg, z)\}$
4. $\{\neg g(y), d(y)\}$
5. $\{\neg f(x,y), \neg f(y,z), f(x,z)\}$
6. $\{h(harry)\}$
7. $\{r(ralph)\}$
8. $\{\neg f(harry, ralph)\}$
9. $\{d(greg)\}$
10. $\{\neg d(y), f(harry, y)\}$
11. $\{f(harry, greg)\}$
12. $\{f(greg, ralph)\}$
13. $\{\neg f(greg, z), f(harry, z)\}$
14. $\{f(harry, ralph)\}$
15. $\{\}$

You need to write justifications in exams!

Example

Given:

$$\exists x.\forall y.(p(x,y) \Leftrightarrow q(x,y))$$
$$\forall x.\exists y.(p(x,y) \vee q(x,y))$$

Prove:

$$\exists x.\exists y.(p(x,y) \wedge q(x,y))$$

Example (continued)

$\exists x.\forall y.(p(x,y) \Leftrightarrow q(x,y))$

$\exists x.\forall y.((\neg p(x,y) \vee q(x,y)) \wedge (p(x,y) \vee \neg q(x,y)))$

$(\neg p(a,y) \vee q(a,y)) \wedge (p(a,y) \vee \neg q(a,y))$

$\{\neg p(a,y), q(a,y)\}$

$\{p(a,y), \neg q(a,y)\}$

$\forall x.\exists y.(p(x,y) \vee q(x,y))$

$p(x, f(x)) \vee q(x, f(x))$

$\{p(a,f(x)), q(a,f(x))\}$

Example (continued)

Negate the goal:

$\exists x.\exists y.(p(x,y) \wedge q(x,y)) \quad \rightarrow \quad \neg\exists x.\exists y.(p(x,y) \wedge q(x,y))$

Convert to Clausal Form:

$\neg\exists x.\exists y.(p(x,y) \wedge q(x,y))$

$\forall x.\forall y.\neg(p(x,y) \wedge q(x,y))$

$\forall x.\forall y.(\neg p(x,y) \vee \neg q(x,y))$

$\neg p(x,y) \vee \neg q(x,y)$

$\{\neg p(x,y), \neg q(x,y)\}$

Example (concluded)

1. $\{\neg p(a,y),\ q(a,y)\}$ Premise
2. $\{p(a,y),\ \neg q(a,y)\}$ Premise
3. $\{p(x, f(x)),\ q(x, f(x))\}$ Premise
4. $\{\neg p(x,y),\ \neg q(x,y)\}$ Negated Goal

5. $\{q(a, f(a))\}$ 1, 3
6. $\{p(a, f(a))\}$ 2, 3
7. $\{\neg p(a,\ f(a))\}$ 4, 5
8. $\{\}$ 6, 7

Try to prove this with S/I rules by yourself!

Problem

Expanded explanation:

Prove $\exists x(p(a, x) \Rightarrow p(x, b))$

(Thanks to Jonathan Borenstein)

$\{ p(a,x) \}$

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

Failure

Neg goal, convert to clauses:

All_x (P(a,x) * ~p(x,b))

P(a,x) * ~p(x,b))

p(a,x)

~p(x,b)

Invalid

Refutation box:

a, b, ~a=b

p(a, b), ~p(b, b),

This make (the empty premises true and) the conclusion false

Relational Resolution II

$$\frac{\{\varphi_1, \ldots, \varphi, \ldots, \varphi_m\} \quad \{\psi_1, \ldots, \neg\psi, \ldots, \psi_n\}}{\{\varphi_1\tau, \ldots, \varphi_m\tau, \psi_1, \ldots, \psi_n\}\sigma}$$

where $\sigma = mgu(\varphi\tau, \psi)$

where τ is a variable renaming

Example

$\{\ p(a,x)\}$

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

Failure

$\{\ p(a,y)\}$

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

$\{\}\{x \leftarrow a, y \leftarrow b\}$

Problem Without Renaming

1. $\{r(a,b,u1)\}$ Premise
2. $\{r(b,c,u2)\}$ Premise
3. $\{r(c,d,u3)\}$ Premise
4. $\{r(x,z,f(v)),\neg r(x,y,f(f(v))),\neg r(y,z,f(f(v)))\}$ Premise
5. $\{\neg r(a,d,w)\}$ Goal
6. $\{r(a,z,f(v)),\neg r(b,z,f(f(v)))\}$ 1,4
7. $\{\neg r(b,d,f(f(v)))\}$ 5,6

8. $\{\neg r(a,y,f(f(v))),\neg r(y,d,f(f(v)))\}$ 4,5
9. $\{\neg r(b,d,f(f(v)))\}$ 1,8

Solution With Renaming

1. $\{r(a,b,u1)\}$ Premise
2. $\{r(b,c,u2)\}$ Premise
3. $\{r(c,d,u3)\}$ Premise
4. $\{r(x,z,f(v)),\neg r(x,y,f(f(v))),\neg r(y,z,f(f(v)))\}$ Premise
5. $\{\neg r(a,d,w)\}$ Goal
6. $\{\neg r(a,y6,f(f(v6))),\neg r(y6,d,f(f(v6)))\}$ 4,5
7. $\{\neg r(b,d,f(f(v7)))\}$ 1,6
8. $\{\neg r(b,y8,f(f(f(v8)))),\neg r(y8,d,f(f(f(v8))))\}$ 4,7
9. $\{\neg r(c,d,f(f(f(v9))))\}$ 2,8
10. $\{\}$ 3,9

Problem

$\{p(x), p(y)\}$

$\{\neg p(u), \neg p(v)\}$

$\{p(y), \neg p(v)\}$

$\{p(x), \neg p(v)\}$

$\{p(y), \neg p(u)\}$

$\{p(x), \neg p(u)\}$

Factors

If a subset of the literals in a clause Φ has a most general unifier γ, then the clause Φ' obtained by applying γ to Φ is called a *factor* of Φ.

Clause

$$\{p(x),p(f(y)),r(x,y)\}$$

Factors

$$\{p(f(y)),r(f(y),y)\}$$

$$\{p(x),p(f(y)),r(x,y)\}$$

Relational Resolution III (Final Version)

Φ

Ψ

$((\Phi' - \{\phi\})\tau \cup (\Psi' - \{\neg\psi\})\tau)\sigma$

where $\phi \in \Phi'$, a factor of Φ

where $\neg\psi \in \Psi'$, a factor of Ψ

where $\sigma = mgu(\phi\tau, \psi)$

where τ is a variable renaming

Example

$\{p(x), p(y)\}$

$\{\neg p(u), \neg p(v)\}$

$\{p(y), \neg p(v)\}$

$\{p(x), \neg p(v)\}$

$\{p(y), \neg p(u)\}$

$\{p(x), \neg p(u)\}$

$\{p(x)\}$

$\{\neg p(u)\}$

$\{\}$

Need for Original Clauses

1. $\{p(a,y), p(x,b)\}$ Premise
2. $\{\neg p(a,d)\}$ Premise
3. $\{\neg p(c,b)\}$ Premise
4. $\{p(x,b)\}$ 1,2
5. $\{\}$ 3,4

1. $\{p(a,y), p(x,b)\}$ Premise
2. $\{\neg p(a,d)\}$ Premise
3. $\{\neg p(c,b)\}$ Premise
4. $\{p(a,b)\}$ Factor of 1

Provability

A *resolution derivation* of a clause ϕ from a set Δ of clauses is a sequence of clauses terminating in ϕ in which each item is
(1) a member of Δ or
(2) the result of applying the resolution to earlier items.

A sentence ϕ is *provable* from a set of sentences Δ by resolution if and only if there is a derivation of the empty clause from the clausal form of $\Delta \cup \{\neg\phi\}$.

A resolution *proof* is a derivation of the empty clause from the clausal form of the premises and the negation of the desired conclusion.

Soundness and Completeness

Metatheorem: Provability using the Relational Resolution Principle is sound and complete for Relational Logic (without equality).