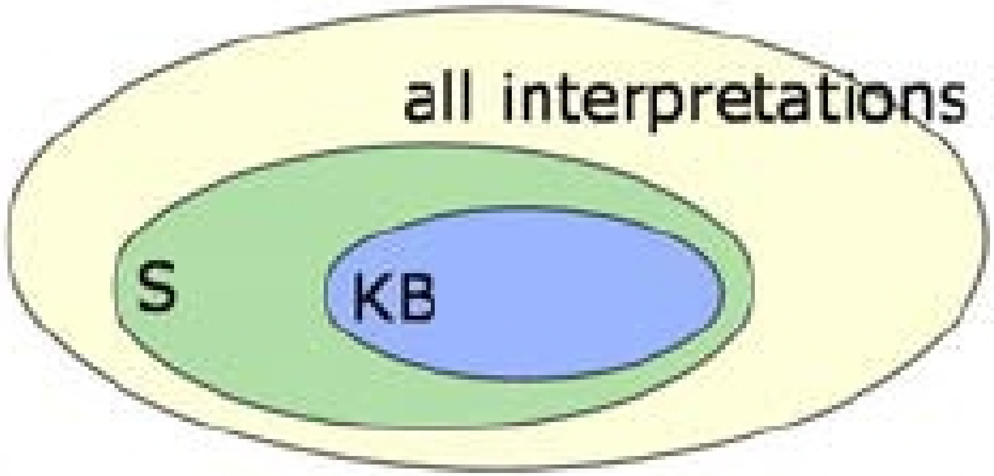
Chapter 9
Inference in First Order Logic

CS 461 – Artificial Intelligence
Pinar Duygulu
Bilkent University, Spring 2008

Slides are mostly adapted from AIMA and MIT Open Courseware

Entailment in First Order Logic

- KB entails S: for every interpretation I, if KB holds in I, then S holds in I



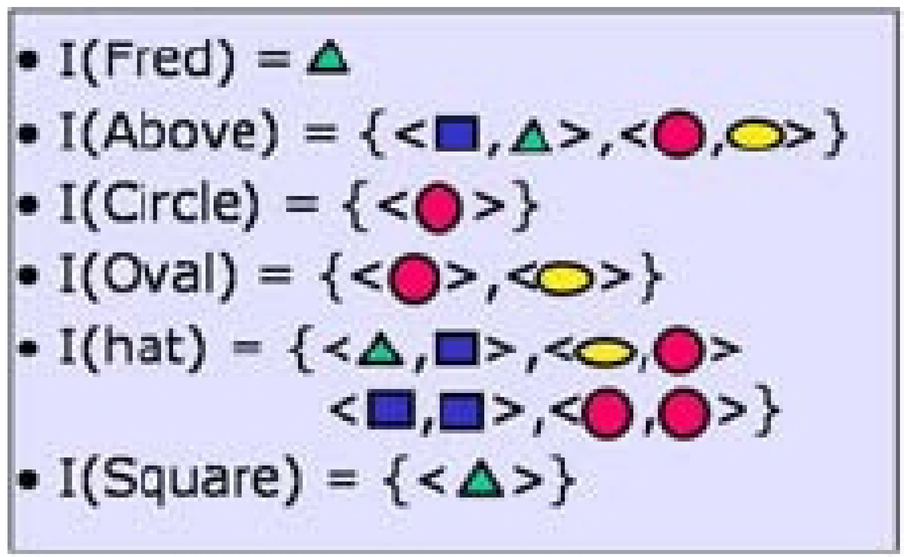
- Computing entailment is impossible in general, because there are infinitely many possible interpretations
- Even computing holds is impossible for interpretations with infinite universes

Intended Interpretations

$KB : (\forall x.\text{Circle}(x) \rightarrow \text{Oval}(x)) \wedge (\forall x.\text{Square}(x) \rightarrow \neg\text{Oval}(x))$

$S : \forall x.\text{Square}(x) \rightarrow \neg\text{Oval}(x)$

- We know holds(KB, I)
- We wonder whether holds(S, I)
- We could ask: Does KB entail S?
- Or we could just try to check whether holds(S, I)



An Infinite Interpretation

$$KB : (\forall x. Circle(x) \rightarrow Oval(x)) \wedge (\forall x. Square(x) \rightarrow \neg Oval(x))$$

$$S : \forall x. Square(x) \rightarrow \neg Oval(x)$$

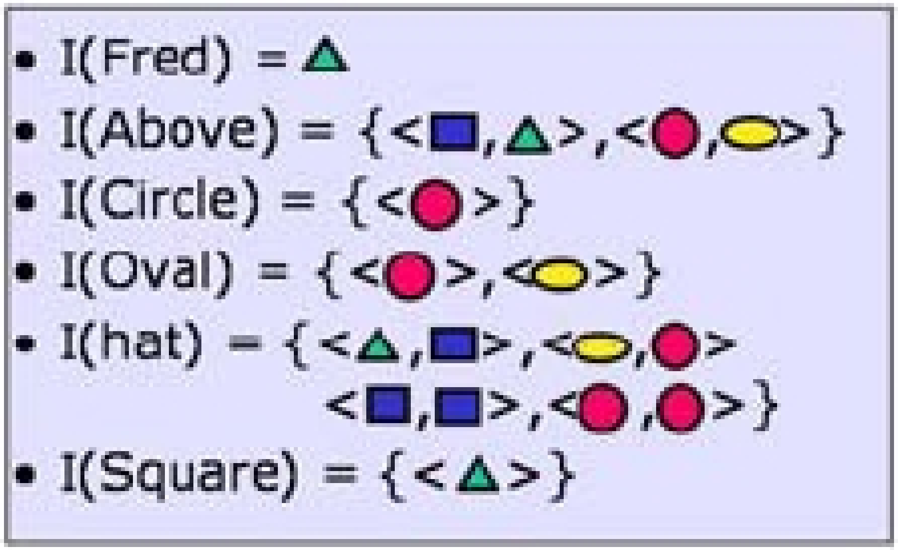
- Does KB hold in I_1?
- Yes, but can't answer via enumerating U
- S also holds in I_1
- No way to verify mechanically

$U_1 = \{1, 2, 3, \ldots\}$
$I_1(circle) = \{4, 8, 12, 16, \ldots\}$
$I_1(oval) = \{2, 4, 6, 8, \ldots\}$
$I_1(square) = \{1, 3, 5, 7, \ldots\}$

An Argument for Entailment

$$KB : (\forall x.\,Circle(x) \rightarrow Oval(x)) \wedge (\forall x.\,Square(x) \rightarrow \neg Oval(x))$$

$$S_1 : \forall x, y.\ Circle(x) \wedge Oval(y) \wedge \neg Circle(y) \rightarrow Above(x, y)$$



$U_1 = \{1, 2, 3, \ldots\}$
$I_1(Circle) = \{4, 8, 12, 16, \ldots\}$
$I_1(Oval) = \{2, 4, 6, 8, \ldots\}$
$I_1(Square) = \{1, 3, 5, 7, \ldots\}$
$I_1(Above) = >$

- holds(KB, I)
- holds(S_1, I)

- holds(KB, I_1)
- fails(S_1, I_1)

KB doesn't entail S_1!

Proof and Entailment

- Entailment captures general notion of "follows from"
- Can't evaluate it directly by enumerating interpretations
- So, we'll do proofs
- In FOL, if S is entailed by KB, then there is a finite proof of S from KB

Axiomatization

- What if we have a particular interpretation, I, in mind, and want to test whether holds(S, I)?
- Write down a set of sentences, called *axioms*, that will serve as our KB
- We would like KB to hold in I, and as few other interpretations as possible
- No matter what,
  - If holds(KB, I) and KB entails S,
  - then holds(S, I)
- If your axioms are weak, it might be that
  - holds(KB, I) and holds(S, I), but
  - KB doesn't entail S

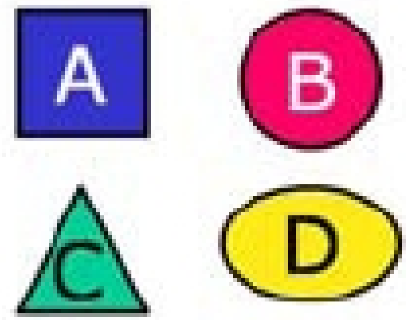
Axiomatization Example

$\text{Above}(A, C)$ KB$_2$

$\text{Above}(B, D)$

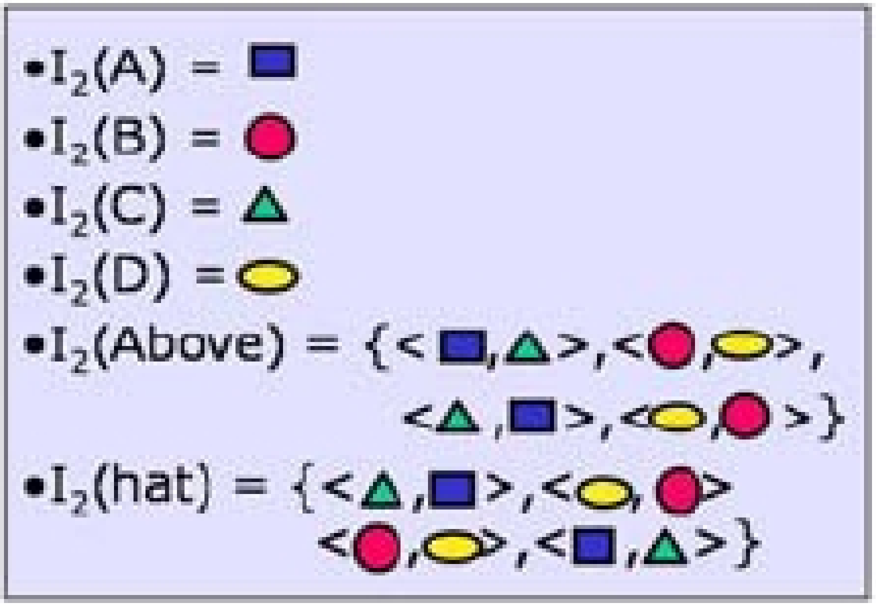
$\forall x, y.\ \text{Above}(x, y) \rightarrow \text{hat}(y) = x$

$\forall x.\ (\neg \exists y.\ \text{Above}(y, x)) \rightarrow \text{hat}(x) = x$

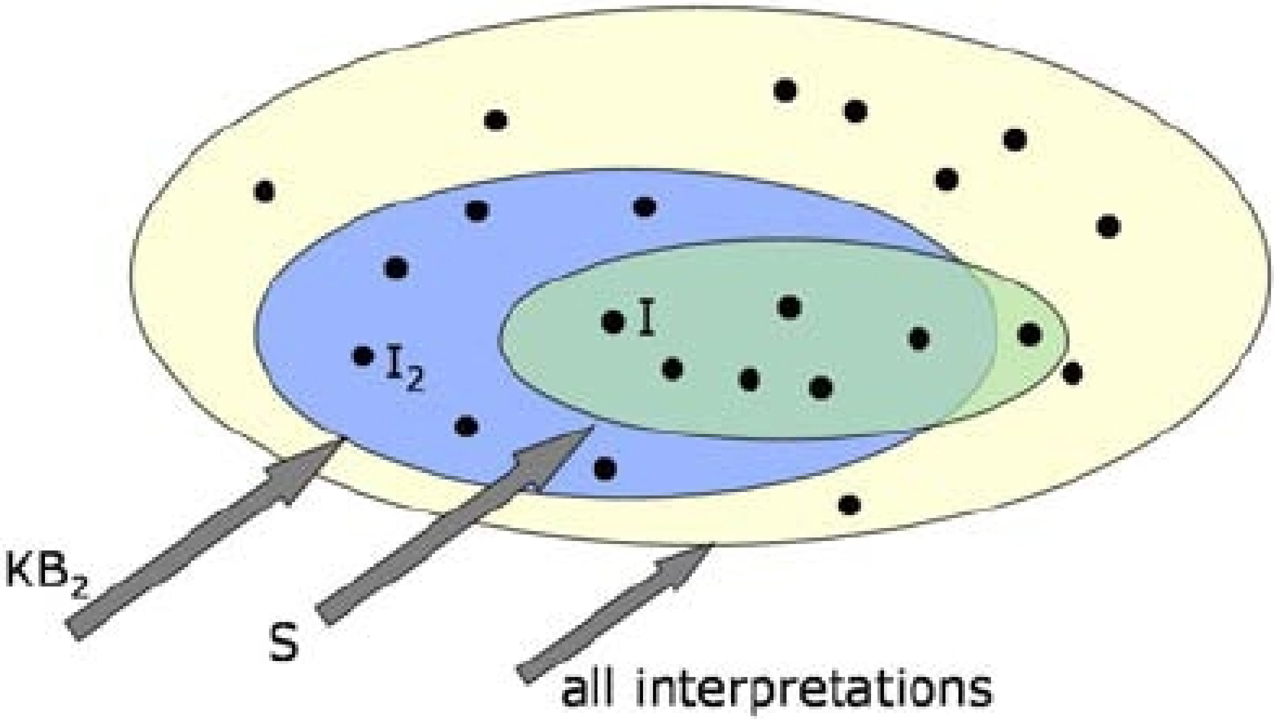


S $\text{hat}(A) = A$

- holds(KB$_2$, I$_2$)
- fails(S, I$_2$)
- KB$_2$ doesn't entail S



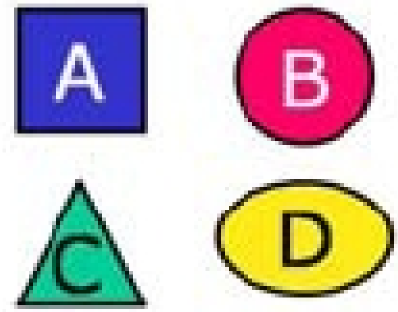
KB2 is a Weakling!



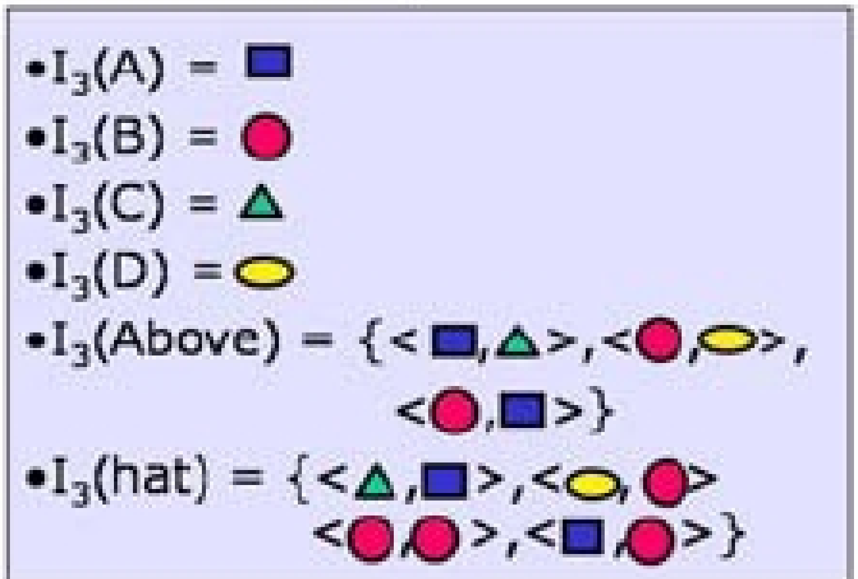
Axiomatization Example: Another Try

KB_3

$$\text{Above}(A, C)$$
$$\text{Above}(B, D)$$
$$\forall x, y.\ \text{Above}(x, y) \rightarrow \text{hat}(y) = x$$
$$\forall x.\ (\neg \exists y.\ \text{Above}(y, x)) \rightarrow \text{hat}(x) = x$$
$$\forall x, y.\ \text{Above}(x, y) \rightarrow \neg \text{Above}(y, x)$$



S: $\text{hat}(A) = A$



- fails(KB_3, I_2)
- holds(KB_3, I_3)
- fails(S, I_3)
- KB_3 doesn't entail S

Axiomatization Example: One last time

KB_4

$Above(A, C)$

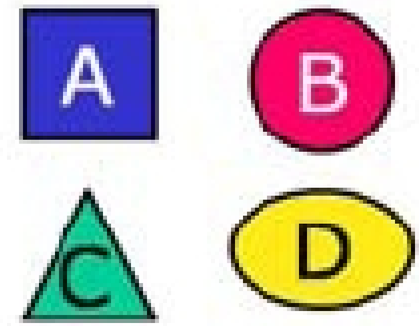
$Above(B, D)$

$\neg\exists x.\ Above(x, A)$

$\neg\exists x.\ Above(x, B)$

$\forall x, y.\ Above(x, y) \rightarrow hat(y) = x$

$\forall x.\ (\neg\exists y.\ Above(y, x)) \rightarrow hat(x) = x$



S $hat(A) = A$

- fails(KB_4, I_3)
- KB_4 entails S

We'll prove S from KB_4 later.

First Order Resolution

$\forall x.\ P(x) \rightarrow Q(x)$

$P(A)$

$Q(A)$

Syllogism:

All men are mortal

Socrates is a man

Socrates is mortal

uppercase letters: constants

lowercase letters: variables

$\forall x.\ \neg P(x) \vee Q(x)$

$P(A)$

$Q(A)$

Equivalent by definition of implication

Two new things:

- converting FOL to clausal form
- resolution with variable substitution

$\neg P(A) \vee Q(A)$

$P(A)$

$Q(A)$

Substitute A for x, still true

then

Propositional resolution

6.034 – Spring 03 • 6

Clausal Form

- like CNF in outer structure
- no quantifiers

$$\forall x.\ \exists y.\ P(x) \rightarrow R(x, y)$$

$$\neg P(x) \vee R(x, F(x))$$

Converting to Clausal Form

1. Eliminate arrows

$$\alpha \leftrightarrow \beta \Rightarrow (\alpha \rightarrow \beta) \wedge (\beta \rightarrow \alpha)$$

$$\alpha \rightarrow \beta \Rightarrow \neg\alpha \vee \beta$$

2. Drive in negation

$$\neg(\alpha \vee \beta) \Rightarrow \neg\alpha \wedge \neg\beta$$

$$\neg(\alpha \wedge \beta) \Rightarrow \neg\alpha \vee \neg\beta$$

$$\neg\neg\alpha \Rightarrow \alpha$$

$$\neg\forall x.\ \alpha \Rightarrow \exists x.\ \neg\alpha$$

$$\neg\exists x.\ \alpha \Rightarrow \forall x.\ \neg\alpha$$

3. Rename variables apart

$$\forall x.\ \exists y.\ (\neg P(x) \vee \exists x.\ Q(x, y)) \Rightarrow$$

$$\forall x_1.\ \exists y_2.\ (\neg P(x_1) \vee \exists x_3.\ Q(x_3, y_2))$$

Converting to Clausal Form - Slolemization

4. Skolemize
   - substitute new name for each existential var

$$\exists x.\, P(x) \Rightarrow P(\mathrm{Fred})$$

$$\exists x, y.\, R(x, y) \Rightarrow R(\mathrm{Thing1}, \mathrm{Thing2})$$

$$\exists x.\, P(x) \wedge Q(x) \Rightarrow P(\mathrm{Fleep}) \wedge Q(\mathrm{Fleep})$$

$$\exists x.\, P(x) \wedge \exists x.\, Q(x) \Rightarrow P(\mathrm{Frog}) \wedge Q(\mathrm{Grog})$$

$$\exists y.\, \forall x.\, \mathrm{Loves}(x, y) \Rightarrow \forall x.\, \mathrm{Loves}(x, \mathrm{Englebert})$$

   - substitute new function of all universal vars in outer scopes

$$\forall x.\, \exists y.\, \mathrm{Loves}(x, y) \Rightarrow \forall x.\, \mathrm{Loves}(x, \mathrm{Beloved}(x))$$

$$\forall x.\, \exists y.\, \forall z.\, \exists w.\, P(x, y, z) \wedge R(y, z, w) \Rightarrow P(x, F(x), z) \wedge R(F(x), z, G(x, z))$$

Converting to Clausal Form

5. Drop universal quantifiers

$$\forall x.\ \text{Loves}(x, \text{Beloved}(x)) \Rightarrow \text{Loves}(x, \text{Beloved}(x))$$

6. Distribute or over and; return clauses

$$P(z) \vee (Q(z,w) \wedge R(w,z)) \Rightarrow \{\{P(z), Q(z,w)\}, \{P(z), R(w,z)\}\}$$

7. Rename the variables in each clause

$$\{\{P(z), Q(z,w)\}, \{P(z), R(w,z)\}\} \Rightarrow \{\{P(z_1), Q(z_1,w_1)\}, \{P(z_2), R(w_2,z_2)\}\}$$

Example

a. John owns a dog
$\exists x.\ D(x) \land O(J,x)$
$D(Fido) \land O(J, Fido)$

b. Anyone who owns a dog is a lover-of-animals
$\forall x.\ (\exists y.\ D(y) \land O(x,y)) \rightarrow L(x)$
$\forall x.\ (\neg\exists y.\ (D(y) \land O(x,y)) \lor L(x)$
$\forall x.\ \forall y.\ \neg(D(y) \land O(x,y)) \lor L(x)$
$\forall x.\ \forall y.\ \neg D(y) \lor \neg O(x,y) \lor L(x)$
$\neg D(y) \lor \neg O(x,y) \lor L(x)$

c. Lovers-of-animals do not kill animals
$\forall x.\ L(x) \rightarrow (\forall y.\ A(y) \rightarrow \neg K(x,y))$
$\forall x.\ \neg L(x) \lor (\forall y.\ A(y) \rightarrow \neg K(x,y))$
$\forall x.\ \neg L(x) \lor (\forall y.\ \neg A(y) \lor \neg K(x,y))$
$\neg L(x) \lor \neg A(y) \lor \neg K(x,y)$

More examples

d. Either Jack killed Tuna or curiosity killed Tuna
K(J,T) v K(C,T)

e. Tuna is a cat
C(T)

f. All cats are animals
¬ C(x) v A(x)

First Order Resolution

$$\frac{\forall x.\ P(x) \rightarrow Q(x) \qquad P(A)}{Q(A)}$$

Syllogism:

All men are mortal
Socrates is a man
Socrates is mortal

uppercase letters: constants

lowercase letters: variables

$$\frac{\forall x.\ \neg P(x) \lor Q(x) \qquad P(A)}{Q(A)}$$

Equivalent by definition of implication

$$\frac{\neg P(A) \lor Q(A) \qquad P(A)}{Q(A)}$$

Substitute A for x, still true

then

Propositional resolution

The key is finding the correct substitutions for the variables.

6.034 - Spring 03 • 1

Substitutions

P(x, f(y), B) : an atomic sentence

Substitution instances	Substitution $\{v_1/t_1, \ldots, v_n/t_n\}$	Comment
P(z, f(w), B)	{x/z, y/w}	Alphabetic variant
P(x, f(A), B)	{y/A}	
P(g(z), f(A), B)	{x/g(z), y/A}	
P(C, f(A), B)	{x/C, y/A}	Ground instance

Applying a substitution:

P(x, f(y), B) {y/A} = P(x,f(A),B)

P(x, f(y), B) {y/A, x/y} = P(A, f(A), B)

Unification

- Expressions ω_1 and ω_2 are unifiable iff there exists a substitution s such that ω_1 s = ω_2 s
- Let ω_1 = x and ω_2 = y, the following are unifiers

s	ω_1 s	ω_2 s
{y/x}	x	x
{x/y}	y	y
{x/f(f(A)), y/f(f(A))}	f(f(A))	f(f(A))
{x/A, y/A}	A	A

Most General Unifier

g is a most general unifier of ω_1 and ω_2 iff for all unifiers s, there exists s′ such that ω_1 s = (ω_1 g) s′ and ω_2 s = (ω_2 g) s′

ω_1	ω_2	MGU
P(x)	P(A)	{x/A}
P(f(x), y, g(x))	P(f(x), x, g(x))	{y/x} or {x/y}
P(f(x), y, g(y))	P(f(x), z, g(x))	{y/x, z/x}
P(x, B, B)	P(A, y, z)	{x/A, y/B, z/B}
P(g(f(v)), g(u))	P(x, x)	{x/g(f(v)), u/f(v)}
P(x, f(x))	P(x, x)	No MGU!

Unification Algorithm

```
unify(Expr x, Expr y, Subst s){
  if s = fail, return fail
  else if x = y, return s
  else if x is a variable, return unify-var(x, y, s)
  else if y is a variable, return unify-var(y, x, s)
  else if x is a predicate or function application,
       if y has the same operator,
            return unify(args(x), args(y), s)
       else  return fail
  else                          ; x and y have to be lists
       return unify(rest(x), rest(y),
                    unify(first(x), first(y), s))
}
```

Unify-var subroutine

Substitute in for var and x as long as possible, then add new binding

```
unify-var(Variable var, Expr x, Subst s){
  if var is bound to val in s,
          return unify(val, x, s)
  else if x is bound to val in s,
          return unify-var(var, val, s)
  else if var occurs anywhere in (x s), return fail
  else return add({var/x}, s)
}
```

Examples

ω_1	ω_2	MGU
A(B, C)	A(x, y)	{x/B, y/C}
A(x, f(D,x))	A(E, f(D,y))	{x/E, y/E}
A(x, y)	A(f(C,y), z)	{x/f(C,y),y/z}
P(A, x, f(g(y)))	P(y, f(z), f(z))	{y/A,x/f(z),z/g(y)}
P(x, g(f(A)), f(x))	P(f(y), z, y)	none
P(x, f(y))	P(z, g(w))	none

Resolution with Variables

$$\frac{\alpha \vee \varphi \qquad \neg\varphi \vee \beta}{(\alpha \vee \beta)\theta} \qquad \mathrm{MGU}(\varphi, \psi) = \theta$$

$$\begin{array}{rl} \forall x, y. & P(x) \vee Q(x, y) \\ \forall x. & \neg P(A) \vee R(B, x) \\ \hline \end{array}$$

$$\begin{array}{rl} \forall x, y. & P(x) \vee Q(x, y) \\ \forall z. & \neg P(A) \vee R(B, z) \\ \hline & (Q(x, y) \vee R(B, z))\theta \\ & Q(A, y) \vee R(B, z) \end{array}$$

$\theta = \{x/A\}$

$$\begin{array}{c} P(x_1) \vee Q(x_1, y_1) \\ \neg P(A) \vee R(B, x_2) \\ \hline (Q(x_1, y_1) \vee R(B, x_2))\theta \\ Q(A, y_1) \vee R(B, x_2) \end{array}$$

$\theta = \{x_1/A\}$

Curiosity Killed the Cat

1	D(Fido)	a
2	O(J,Fido)	a
3	¬ D(y) v ¬ O(x,y) v L(x)	b
4	¬ L(x) v ¬ A(y) v ¬ K(x,y)	c
5	K(J,T) v K(C,T)	d
6	C(T)	e
7	¬ C(x) v A(x)	f
8	¬ K(C,T)	Neg
9	K(J,T)	5,8
10	A(T)	6,7 {x/T}
11	¬ L(J) v ¬ A(T)	4,9 {x/J, y/T}
12	¬ L(J)	10,11
13	¬ D(y) v ¬ O(J,y)	3,12 {x/J}
14	¬ D(Fido)	13,2 {y/Fido}
15	▪	14,1

Proving Validity

- How do we use resolution refutation to prove something is valid?
- Normally, we prove a sentence is entailed by the set of axioms
- Valid sentences are entailed by the empty set of sentences
- To prove validity by refutation, negate the sentence and try to derive contradiction.

Example

- Syllogism

$$\left(\forall x. P(x) \rightarrow Q(x)\right) \wedge P(A) \rightarrow Q(A)$$

- Negate and convert to clausal form

$$\neg\left(\left(\forall x. P(x) \rightarrow Q(x)\right) \wedge P(A) \rightarrow Q(A)\right)$$

$$\neg\left(\neg\left(\forall x. \neg P(x) \vee Q(x)\right) \vee \neg P(A) \vee Q(A)\right)$$

$$\left(\forall x. \neg P(x) \vee Q(x)\right) \wedge P(A) \wedge \neg Q(A)$$

$$\left(\neg P(x) \vee Q(x)\right) \wedge P(A) \wedge \neg Q(A)$$

Example

- Do proof

1.	$\neg P(x) \vee Q(x)$	
2.	$P(A)$	
3.	$\neg Q(A)$	
4.	$Q(A)$	1,2
5.	■	3,4

Green's Trick

- Use resolution to get answers to existential queries

$$\exists x.\ Mortal(x)$$

1.	$\neg Man(x) \vee Mortal(x)$	
2.	$Man(Socrates)$	
3.	$\neg Mortal(x) \vee Answer(x)$	
4.	$Mortal(Socrates)$	1,2
5.	$Answer(Socrates)$	3,5

Equality

- Special predicate in syntax and semantics; need to add something to our proof system
- Could add another special inference rule called paramodulation
- Instead, we will axiomatize equality as an equivalence relation

$$\forall x. Eq(x, x)$$

$$\forall x, y. Eq(x, y) \rightarrow Eq(y, x)$$

$$\forall x, y, z. Eq(x, y) \wedge Eq(y, z) \rightarrow Eq(x, z)$$

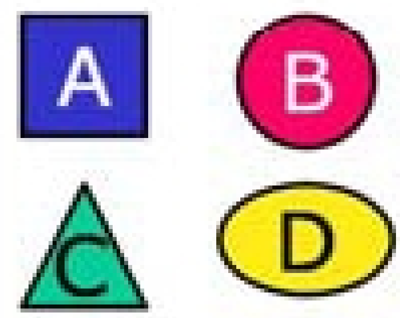
- For every predicate, allow substitutions

$$\forall x, y. Eq(x, y) \rightarrow (P(x) \rightarrow P(y))$$

Proof Example

- Let's go back to our old geometry domain and try to prove what the hat of A is
- Axioms in FOL (plus equality axioms)

$$\mathrm{Above}(A, C)$$
$$\mathrm{Above}(B, D)$$
$$\neg\exists x.\ \mathrm{Above}(x, A)$$
$$\neg\exists x.\ \mathrm{Above}(x, B)$$
$$\forall x, y.\ \mathrm{Above}(x, y) \rightarrow \mathrm{hat}(y) = x$$
$$\forall x.\ (\neg\exists y.\ \mathrm{Above}(y, x)) \rightarrow \mathrm{hat}(x) = x$$



- Desired conclusion: $\exists x.\ \mathrm{hat}(A) = x$
- Use Green's trick to get the binding of x

The Clauses

1.	Above(A, C)	
2.	Above(B, D)	
3.	~Above(x, A)	
4.	~Above(x, B)	
5.	~Above(x, y) v Eq(hat(y), x)	
6.	Above(sk(x), x) v Eq(hat(x), x)	
7.	Eq(x, x)	
8.	~Eq(x, y) v ~Eq(y, z) v Eq(x, z)	
9.	~Eq(x, y) v Eq(y, x)	
10.		
11.		
12.		

The Query

1.	Above(A, C)	
2.	Above(B, D)	
3.	~Above(x, A)	
4.	~Above(x, B)	
5.	~Above(x, y) v Eq(hat(y), x)	
6.	Above(sk(x), x) v Eq(hat(x), x)	
7.	Eq(x, x)	
8.	~Eq(x, y) v ~Eq(y, z) v Eq(x, z)	
9.	~Eq(x, y) v Eq(y, x)	
10.	~Eq(hat(A), x) v Answer(x)	

The Proof

1.	Above(A, C)	
2.	Above(B, D)	
3.	~Above(x, A)	
4.	~Above(x, B)	
5.	~Above(x, y) v Eq(hat(y), x)	
6.	Above(sk(x), x) v Eq(hat(x), x)	
7.	Eq(x, x)	
8.	~Eq(x, y) v ~Eq(y, z) v Eq(x, z)	
9.	~Eq(x, y) v Eq(y, x)	
10.	~Eq(hat(A), x) v Answer(x)	conclusion
11.	Above(sk(A), A) v Answer(A)	6, 10 {x/A}
12.	Answer(A)	11, 3 {x/sk(A)}

Hat of D

1.	Above(A, C)	
2.	Above(B, D)	
3.	~Above(x, A)	
4.	~Above(x, B)	
5.	~Above(x, y) v Eq(hat(y), x)	
6.	Above(sk(x), x) v Eq(hat(x), x)	
7.	Eq(x, x)	
8.	~Eq(x, y) v ~Eq(y, z) v Eq(x, z)	
9.	~Eq(x, y) v Eq(y, x)	
10.	~Eq(hat(D), x) v Answer(x)	conclusion
11.	~Above(x,D) v Answer(x)	5, 10 {x1/x}
12.	Answer(B)	11, 2 {x/B}

Who is Jane's Lower

- Jane's lover drives a red car
- Fred is the only person who drives a red car
- Who is Jane's lover?

1.	Drives(lover(Jane))	
2.	~Drives(x) v Eq(x,Frec)	
3.	~Eq(lover(Jane),x) v Answer(x)	
4.	Eq(lover(Jane), Fred)	1,2 {x/lover(Jane)}
5.	Answer(Fred)	3,4 {x/Fred}