

# Inference in First-Order Logic.

Chapters 8 & 9

# Announcements

- A2 due tonight 11:59pm
- A3 is out
- A1 is corrected and solutions available in class or from Yuyang

# Last time: First-Order Logic

- Syntax:
  - **terms** (constants, variables, function symbols)
  - sentences: **predicates** on one or more terms, perhaps connected with  $\neg, \wedge, \vee, \Rightarrow$  or  $\Leftrightarrow$ , perhaps quantified with a  $\forall$  or  $\exists$
- Semantics:
  - a model (collection of objects and relations on objects)
  - an **interpretation** (assignment of symbols to objects and relations)
  - a predicate is true in a model under an interpretation iff the corresponding relation is true between the corresponding objects in the model, under the interpretation
  - existential quantifier on predicate is true iff at least one object obeys the relation
  - universal quantifier true iff all objects obey the relation

# Today: FOL Inference

- How to use FOL in KBs
  - ASK, TELL, substitution of variables
- Inference in FOL
  - unification
- Resolution in FOL
  - theorem-proving
- KBs made of rules and inference with (Generalized) Modus Ponens
  - by forward and backward chaining

# Interacting with FOL KBs

- Suppose a wumpus-world agent is using a FOL KB and perceives a smell and a breeze (but no glitter) at  $t=5$ :  
 $TELL(KB, Percept([Smell, Breeze, None], 5))$   
 $ASK(KB, \exists a BestAction(a, 5))$
- I.e., does the KB entail some best action at  $t=5$ ?
- Answer: Yes,  $\{a/Shoot\}$  ← substitution (binding list)
- Given a sentence  $S$  and a substitution  $\theta$ ,
- $SUBST(S, \theta)$  denotes the result of plugging  $\theta$  into  $S$ ; e.g.,  
 $S = Smarter(x, y)$   
 $\theta = \{x/Hillary, y/Bill\}$   
 $SUBST(S, \theta) = Smarter(Hillary, Bill)$
- $ASK(KB, S)$  returns some/all  $\theta$  such that  $KB \models SUBST(S, \theta)$

# FOL resolution: example

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

All men are mortal  
Socrates is a man

constants  
are uppercase

$$Q(A)$$

Socrates is mortal

variables  
are lowercase

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

Equivalent by definition  
of implication

1. Clausal form

$$\frac{\neg P(A) \vee Q(A)}{P(A)}$$

Substitute  $A$  for  $x$   
(still true)

2. Automatic  
substitution

$$Q(A)$$

Propositional resolution

## Clausal form

- Like CNF in outer structure
  - a conjunction of clauses (disjunctions)
- No quantifiers

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

↓

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

Mon, Sept 29

COMP 131, Lecture 8

7

## Conversion to clausal form

- Eliminate equivalences and implications (arrows):
  - $a \Leftrightarrow b$  becomes  $(a \Rightarrow b) \wedge (b \Rightarrow a)$
  - $a \Rightarrow b$  becomes  $\neg a \vee b$
- Drive in negation (de Morgan's rules) and remove double negation:
  - $\neg(a \wedge b)$  becomes  $(\neg a \vee \neg b)$
  - $\neg(a \vee b)$  becomes  $(\neg a \wedge \neg b)$
  - $\neg\neg a$  becomes  $a$
  - $\neg\forall x. a$  becomes  $\exists x. \neg a$
  - $\neg\exists x. a$  becomes  $\forall x. \neg a$
- Rename all variables with different names (standardize apart):
  - $\forall x. \exists y. \neg P(x) \vee \exists x. Q(x, y)$  becomes  $\forall x_1. \exists y_1. \neg P(x_1) \vee \exists x_2. Q(x_2, y_1)$

Mon, Sept 29

COMP 131, Lecture 8

8

## Skolemization

### 4. Skolemize:

- substitute a *new* constant symbol for every existential var

Symbol not found elsewhere in the KB

$$\begin{aligned} \exists x. P(x) &\text{ becomes } P(\text{Fred}) \\ \exists x, y. R(x, y) &\text{ becomes } R(\text{Thing1}, \text{Thing2}) \\ \exists x. P(x) \wedge Q(x) &\text{ becomes } P(A) \wedge Q(A) \\ \exists x. P(x) \wedge \exists x. Q(x) &\text{ becomes } P(A) \wedge Q(B) \\ \forall x \exists y. Loves(x, y) &\text{ becomes } \forall x. Loves(x, \text{Raymond}) \end{aligned}$$

- substitute a new function for every universal var in outer scope

$$\begin{aligned} \forall x \exists y. Loves(x, y) &\text{ becomes } \forall x. Loves(x, \text{Beloved}(x)) \\ \forall x \exists y \forall z \exists w. P(x, y, z) \vee Q(y, z, w) &\text{ becomes } \\ &\forall x \forall z. P(x, F(x), z) \vee Q(F(x), z, G(x, z)) \end{aligned}$$

Mon, Sept 29

COMP 131, Lecture 8

9

## Convert to clausal form

### 5. Drop universal quantifiers:

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

### 6. Distribute or over and; return clauses

$$P(x) \vee (Q(x, y) \wedge R(z, y)) \text{ becomes } \{ \{P(x), Q(x, y)\} \{P(x), R(z, y)\} \}$$

### 7. Rename variables to different symbols in different clauses

- already done in step 3

$$\{ \{P(x), Q(x, y)\} \{P(x), R(z, y)\} \} \text{ becomes } \{ \{P(x_1), Q(x_1, y_1)\} \{P(x_2), R(z_2, y_2)\} \}$$

axioms

DONE!  
KB in  
clausal form

Mon, Sept 29

COMP 131, Lecture 8

10

## Conversion examples

### 1. Jack has a dog.

$$\begin{aligned} \exists x. Dog(x) \wedge Has(J, x) &\quad \text{skolemize} \\ Dog(\text{Fido}) \wedge Has(J, \text{Fido}) & \\ \{ \{Dog(\text{Fido})\} \{Has(J, \text{Fido})\} \} & \end{aligned}$$

### 2. Anyone who has a dog is a lover-of-animals.

$$\begin{aligned} \forall x, y. Has(x, y) \wedge Dog(y) \Rightarrow LoA(x) &\quad \text{remove implication} \\ \forall x, y. \neg(Has(x, y) \wedge Dog(y)) \vee LoA(x) &\quad \text{drive negation in} \\ \forall x, y. \neg Has(x, y) \vee \neg Dog(y) \vee LoA(x) &\quad \text{drop universals} \\ \neg Has(x, y) \vee \neg Dog(y) \vee LoA(x) & \\ \{ \{ \neg Has(x, y), \neg Dog(y), LoA(x) \} \} & \end{aligned}$$

Mon, Sept 29

COMP 131, Lecture 8

11

## More conversion

### 3. Lovers-of-animals don't kill animals

$$\begin{aligned} \forall x. LoA(x) \Rightarrow (\forall y. Animal(y) \Rightarrow \neg Kills(x, y)) & \\ \forall x. \neg LoA(x) \vee (\forall y. \neg Animal(y) \vee \neg Kills(x, y)) & \\ \neg LoA(x) \vee \neg Animal(y) \vee \neg Kills(x, y) & \\ \{ \{ \neg LoA(x), \neg Animal(y), \neg Kills(x, y) \} \} & \end{aligned}$$

Mon, Sept 29

COMP 131, Lecture 8

12

## Automatic substitution

- We have a KB made of atomic sentences (no quantifiers) in clausal form
- Still have variables
- Substitution rules
  - can make an "alphabetic variant" substitution  
 $SUBST(Kills(x,y), \{x/z, y/w\}) = Kills(z,w)$
  - can substitute a ground term for a variable (**universal instantiation**)  
 $SUBST(Kills(x,y), \{x/Jack, y/Mother(Fido)\}) = Kills(Jack, Mother(Fido))$

Mon, Sept 29

COMP 131, Lecture 8

13

## Unification

- A substitution process that makes two expressions match each other exactly
  - Expressions  $\alpha_1$  and  $\alpha_2$  are **unifiable** iff there exists a substitution  $s$  s.t.  $SUBST(\alpha_1, s) = SUBST(\alpha_2, s)$
- Some unifiers for  $\alpha_1 = x$  and  $\alpha_2 = y$

substitution $s$	$SUBST(\alpha_1, s)$	$SUBST(\alpha_2, s)$
$x/y$	$y$	$y$
$y/x$	$x$	$x$
$x/F(x), y/F(x)$	$F(x)$	$F(x)$
$x/A, y/A$	$A$	$A$

Mon, Sept 29

COMP 131, Lecture 8

14

## Most General Unifier (MGU)

- $g$  is the MGU for  $\alpha_1$  and  $\alpha_2$  iff for all unifiers  $s$  there is a  $s'$  s.t.  
 $SUBST(\alpha, s) = SUBST(SUBST(\alpha, g), s')$
- A unifier is most general iff every other unifier can be expressed as another substitution after the most general one
- To make a MGU, make the *fewest possible commitments* and still make the two expressions equal

Mon, Sept 29

COMP 131, Lecture 8

15

## MGU examples

$\alpha_1$	$\alpha_2$	MGU
$P(x)$	$P(A)$	$x/A$
$R(F(x), y, G(x))$	$R(F(x), x, G(x))$	$x/y$ or $y/x$
$R(F(x), y, G(x))$	$R(F(x), z, G(x))$	$y/x, z/x$
$R(x, A, B)$	$R(C, y, z)$	$x/C, y/A, z/B$
$Q(G(F(w)), G(u))$	$Q(x, x)$	$x/G(F(w)), u/F(w)$
$P(x, F(x))$	$P(x, x)$	No MGU!

can't substitute a variable for a function because the range of the function could be more specific

Mon, Sept 29

COMP 131, Lecture 8

16

## MGU computing algorithm

```

function UNIFY(x, y,  $\theta$ ) returns a substitution to make x and y identical
inputs: x, a variable, constant, list, or compound
       y, a variable, constant, list, or compound
        $\theta$ , the substitution built up so far

if  $\theta = \text{failure}$  then return failure
else if  $x = y$  then return  $\theta$ 
else if VARIABLE?(x) then return UNIFY-VAR(x, y,  $\theta$ )
else if VARIABLE?(y) then return UNIFY-VAR(y, x,  $\theta$ )
else if COMPOUND?(x) and COMPOUND?(y) then
    return UNIFY(ARGS[x], ARGS[y], UNIFY(OP[x], OP[y],  $\theta$ ))
else if LIST?(x) and LIST?(y) then
    return UNIFY(REST[x], REST[y], UNIFY(FIRST[x], FIRST[y],  $\theta$ ))
else return failure
    
```

Compound (e.g.,  $F(A, B)$ ):  
OP picks out function symbol, ARGS picks out arguments

Mon, Sept 29

COMP 131, Lecture 8

17

## Unification of variables

Substitute  $\theta$  for  $var$  and  $x$  has long as possible, then add new binding

```

function UNIFY-VAR(var, x,  $\theta$ ) returns a substitution
inputs: var, a variable
       x, any expression
        $\theta$ , the substitution built up so far

if {var/val}  $\in \theta$  then return UNIFY(val, x,  $\theta$ )
else if {x/val}  $\in \theta$  then return UNIFY(var, val,  $\theta$ )
else if OCCUR-CHECK?(var, x) then return failure
else return add {var/x} to  $\theta$ 
    
```

OCCUR\_CHECK expensive step, keeps us from circularities like  $x/f(x)$

Mon, Sept 29

COMP 131, Lecture 8

18

## FOL Resolution!

$$\frac{\alpha \vee \varphi \quad \neg \psi \vee \beta, \text{MGU}(\varphi, \psi) = \theta}{\text{SUBST}(\alpha \vee \beta, \theta)}$$

- If there is a MGU substitution *theta* that unifies parts of both clauses s.t. they become complementary, then we can conclude with the result of the substitution *theta* applied to the rest of both clauses

$$\frac{P(x) \vee Q(x,y) \quad \neg P(A) \vee R(B,z)}{\text{SUBST}(Q(x,y) \vee R(B,z), \theta)} \quad \theta = \{x/A\}$$

$$\text{SUBST}(Q(x,y) \vee R(B,z), \theta)$$

$$Q(A,y) \vee R(B,z)$$

Mon, Sept 29

COMP 131, Lecture 8

19

## FOL resolution: example

$$P(x) \vee Q(x,y)$$

$$\neg P(A) \vee R(B,x)$$

$$\frac{P(x_1) \vee Q(x_1,y_1) \quad \neg P(A) \vee R(B,x_2)}{\text{SUBST}(Q(x_1,y_1) \vee R(B,x_1), \theta)}$$

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

$$\text{SUBST}(Q(x_1,y_1) \vee R(B,x_1), \theta)$$

$$Q(A,y_1) \vee R(B,x_2)$$

- What if there's a repeated variable (x)?
- Implicit universal quantifiers!
- Step 7: rename variables to different symbols in different clauses

Mon, Sept 29

COMP 131, Lecture 8

20

## Did curiosity kill the cat?

1	<i>Dog(Fido)</i>	
2	<i>Has(J,Fido)</i>	
3	$\neg \text{Has}(x,y) \vee \neg \text{Dog}(y) \vee \text{LoA}(x)$	
4	$\neg \text{LoA}(x) \vee \neg \text{Animal}(y) \vee \neg \text{Kills}(x,y)$	
5	$\text{Kills}(J,\text{Tuna}) \vee \text{Kills}(C,\text{Tuna})$	
6	<i>Cat(Tuna)</i>	
7	$\neg \text{Cat}(x) \vee \text{Animal}(x)$	
8	$\neg \text{Kills}(C,\text{Tuna})$	Negate conclusion

Mon, Sept 29

COMP 131, Lecture 8

21

## Did curiosity kill the cat?

1	<i>Dog(Fido)</i>		Resolution fails Therefore, KB entails conclusion: Curiosity killed the cat.
2	<i>Has(J,Fido)</i>		
3	$\neg \text{Has}(x,y) \vee \neg \text{Dog}(y) \vee \text{LoA}(x)$		
4	$\neg \text{LoA}(x) \vee \neg \text{Animal}(y) \vee \neg \text{Kills}(x,y)$		
5	$\text{Kills}(J,\text{Tuna}) \vee \text{Kills}(C,\text{Tuna})$		
6	<i>Cat(Tuna)</i>		
7	$\neg \text{Cat}(x) \vee \text{Animal}(x)$		
8	$\neg \text{Kills}(C,\text{Tuna})$	Negate conclusion	
9	$\text{Kills}(J,\text{Tuna})$	5, 8	
10	<i>Animal(Tuna)</i>	6, 7 { <i>x/Tuna</i> }	
11	$\neg \text{LoA}(J) \vee \neg \text{Animal}(Tuna)$	4, 9 { <i>x/J, y/Tuna</i> }	
12	$\neg \text{LoA}(J)$	10, 11	
13	$\neg \text{Has}(J,y) \vee \neg \text{Dog}(y)$	3, 12, { <i>x/J</i> }	
14	$\neg \text{Dog}(Fido)$	13, 2 { <i>y/Fido</i> }	
15	empty clause	14, 1	

Mon, Sept 29

COMP 131, Lecture 8

22

## The power of fail

- For both PL and FOL,  $(\neg P \wedge P)$  entails anything. Why?
  - Prove using resolution

Mon, Sept 29

COMP 131, Lecture 8

23

## Proving validity

- Use resolution to prove that *P* is valid (a tautology)
  - start with an empty KB, negate *P*, derive contradiction

Mon, Sept 29

COMP 131, Lecture 8

24

## Resolution and factoring

- Binary resolution that matches one literal from each clause is not complete
  - can we get a contradiction from these clauses?
 
$$\begin{array}{l} P(x) \vee P(y) \\ \neg P(y) \vee \neg P(w) \end{array}$$
  - we should! (there's no interpretation that makes both true)
  - but all we get is  $P(x) \vee \neg P(w)$
  - and from there, all we can get is one of the original clauses
  - so we found a counter-example for binary resolution

Mon, Sept 29

COMP 131, Lecture 8

25

## Factoring

- Generalized resolution lets you resolve away several literals at once (p.297 of AIMA)
- Simpler to introduce a new inference rule, called **factoring**

$$\frac{\alpha\beta\vee\gamma \quad \theta = \text{MGU}(\alpha,\beta)}{\text{SUBST}(\alpha\vee\gamma, \theta)}$$

because unification makes  $\alpha$  and  $\beta$  equivalent

- Example

$$\begin{array}{l} Q(y) \vee P(x,y) \vee P(v,A) \\ (Q(y) \vee P(x,y)) \{x/v, y/A\} \\ Q(A) \vee P(v,A) \end{array}$$

- Binary resolution plus factoring is complete

Mon, Sept 29

COMP 131, Lecture 8

26

## Green's trick

- Answers to existential queries: e.g.,  $\exists x.Mortal(x)$  ?

1	$\neg Man(x) \vee Mortal(x)$	
2	$Man(Socrates)$	
3	$\neg Mortal(x) \vee Answer(x)$	Neg., or answer
4	$Mortal(Socrates)$	1, 2 $\{x/Socrates\}$
5	$Answer(Socrates)$	3, 4, $\{x/Socrates\}$

remember: existential queries is how we ASK the KB

Mon, Sept 29

COMP 131, Lecture 8

27

## Equality

- Special predicate in syntax and semantics; need to add something to our proof system
- Could add another special inference rule (paramodulation, see p.304 of AIMA if interested)
- Instead, we axiomatize equality as an equivalence relation:

$$\begin{array}{l} \forall x . Eq(x,x) \\ \forall x,y . Eq(x,y) \Rightarrow Eq(y,x) \\ \forall x,y,z . Eq(x,y) \wedge Eq(y,z) \Rightarrow Eq(x,z) \end{array}$$

- For every predicate, allow substitutions:

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

Mon, Sept 29

COMP 131, Lecture 8

28

## Completeness and decidability

- Complete: if KB entails S, then we can prove S from KB
- Gödel's Completeness Theorem (1930): *There exists a complete proof system for FOL*
- Robinson found such a proof system (general resolution) in 1965
- *FOL is semi-decidable*: if the desired conclusion follows from the premises, then refutation by resolution will eventually prove it. But if it doesn't, maybe it will halt, or maybe not.

Mon, Sept 29

COMP 131, Lecture 8

29

## Adding arithmetic

- Gödel's Incompleteness Theorem (1931): *There is no consistent, complete proof system for FOL + arithmetic.*
- Either there are sentences that are true but not provable, or there are sentences that are provable but not true.
- How does that work?
- Arithmetic gives the ability to construct code-names for sentences within the logic
  - $P = \text{"}P \text{ is not provable"}$
  - if  $P$  is true:  $P$  is not provable (incomplete)
  - if  $P$  is false:  $P$  is (not not) provable (inconsistent)

Mon, Sept 29

COMP 131, Lecture 8

30

## Theorem-provers

- Use resolution as part of an algorithm that infers new knowledge from existing KB
- Efficient resolution strategies
  - **unit preference**: resolve clauses where one contains a single literal first
  - **set of support**: every resolution combines one clause from the set of support and one other clause, and adds the result into the set of support
    - E.g., initially set of support = negated conclusion

Mon, Sept 29

COMP 131, Lecture 8

31

## Next time

- Forward and backward chaining
- Knowledge rep with FOL
- Intro to NLP

Mon, Sept 29

COMP 131, Lecture 8

32