

# Knowledge representation and reasoning (KRR)

- First-order logic: syntax and semantics
- Resolution-based inference procedure

# What is KRR?

Symbolic encoding of propositions believed by some agent and their manipulation to produce representations of propositions that are believed by the agent but not explicitly represented

## An example

- Explicitly represented beliefs:

$GradStu(Ann), GradStu(Bob),$   
 $\forall x(GradStu(x) \rightarrow Student(x))$

- Implicitly represented beliefs:

$Student(Ann), Student(Bob),$   
 $\forall x(\neg Student(x) \rightarrow \neg GradStu(x))$

# We need knowledge to answer questions

Could a crocodile run a steeplechase?

[Levesque 88]

- Yes
- No

**The intended thinking:** short legs, tall hedges  $\Rightarrow$  No!

# Yet another example

Consider a question about materials:

The large ball crashed right through the table because it was made of **XYZZY**. What was made of **XYZZY**?

- the large ball
- the table

Now suppose that you learn some facts about **XYZZY**.

1. It is a trademarked product of the Dow Chemical Company.
2. It is usually white, but there are green and blue varieties.
3. It is ninety-eight percent air, making it lightweight and buoyant.
4. It was first discovered by a Swedish inventor, Carl Georg Munters.

**Ask:** At what point does the answer stop being just a guess?

# Why KRR?

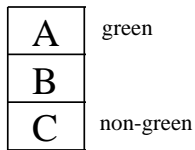
- KR hypothesis: any artificial intelligent system is knowledge-based
- Knowledge-based system: system with structures that
  - can be interpreted propositionally and
  - determine the system behaviorsuch structures are called its knowledge base (KB)
- Knowledge-based system most suitable for open-ended tasks
- Hallmark of knowledge-based system: cognitive penetrability, *i.e.*, actions depend on beliefs, including implicitly represented beliefs

Logic is the main tool for KRR, because logic studies

- How to formally represent agent's beliefs
- Given the explicitly represented beliefs, what are the implicitly represented beliefs

There are many kinds of logics. In this course, we will use first-order logic (FOL) as the tool for KRR

# A blocks world example



- Given the scene, human can easily draw the conclusion “there is a green block directly on top of a non-green block”
- How can a machine do the same?

# Formalization in FOL

A	green
B	
C	non-green

- $S = \{On(a, b), On(b, c), Green(a), \neg Green(c)\}$
- $\alpha = \exists x \exists y [Green(x) \wedge \neg Green(y) \wedge On(x, y)]$
- $S$  logically entails  $\alpha$



# An example

- Tony, Mike, and John belong to the Alpine Club.
- Every member of the Alpine Club who is not a skier is a mountain climber.
- Mountain climbers do not like rain, and anyone who does not like snow is not a skier.
- Mike dislikes whatever Tony likes, and likes whatever Tony dislikes.
- Tony likes rain and snow.
- Is there a member of the Alpine Club who is a mountain climber but not a skier?

# An example (cont'd)

- Intelligence is needed to answer the question
- Can we make machines answer the question?
- A possible approach
  - First, translate the sentences and question into FOL formulas
    - Of course, this is hard, and we do not have a good way to automate this step
  - Second, check if the formula of the question is logically entailed by the formulas of the sentences
    - We will show that there are ways to automate this step

- Individuals (constants or 0-ary functions):
  - tony, mike, john
  - rain, snow
- Types (unary predicates):
  - $A(x)$  means that  $x$  belongs to Alpine Club
  - $S(x)$  means that  $x$  is a skier
  - $C(x)$  means that  $x$  is a mountain climber
- Relationships (binary predicates):
  - $L(x, y)$  means that  $x$  likes  $y$

# Basic facts

- Tony, Mike, and John belong to the Alpine Club.

$A(tony), A(mike), A(john)$

- Tony likes rain and snow.

$L(tony, rain), L(tony, snow)$

# Complex facts

- Every member of the Alpine Club who is not a skier is a mountain climber.

$$\forall x(A(x) \wedge \neg S(x)) \rightarrow C(x)$$

- Mountain climbers do not like rain, and anyone who does not like snow is not a skier.

$$\forall x(C(x) \rightarrow \neg L(x, \text{rain}))$$

$$\forall x(\neg L(x, \text{snow}) \rightarrow \neg S(x))$$

- Mike dislikes whatever Tony likes, and likes whatever Tony dislikes.

$$\forall x(L(\text{tony}, x) \rightarrow \neg L(\text{mike}, x))$$

$$\forall x(\neg L(\text{tony}, x) \rightarrow L(\text{mike}, x))$$

- Is there a member of the Alpine Club who is a mountain climber but not a skier?

$$\exists x(A(x) \wedge C(x) \wedge \neg S(x))$$

# Alphabet

Logical symbols (fixed meaning and use):

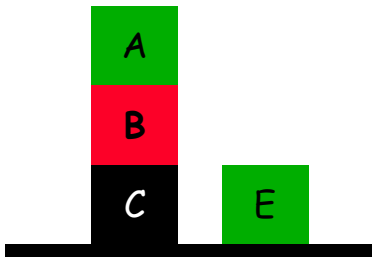
- Punctuation:  $(, ), , , .$
- Connectives and quantifiers:  $=, \neg, \wedge, \vee, \forall, \exists$
- Variables:  $x, x_1, x_2, \dots, x', x'', \dots, y, \dots, z, \dots$

Non-logical symbols (domain-dependent meaning and use):

- Predicate symbols
  - arity: number of arguments
  - arity 0 predicates: propositional symbols
- Function symbols
  - arity 0 functions: constant symbols

# A blocks world example

## Environment



## Language (Syntax)

- **Constants:** a,b,c,e
- **Functions:**
  - No function
- **Predicates:**
  - on: binary
  - above: binary
  - clear: unary
  - ontable: unary

- Every variable is a term
- If  $t_1, \dots, t_n$  are terms and  $f$  is a function symbol of arity  $n$ , then  $f(t_1, \dots, t_n)$  is a term



- If  $t_1, \dots, t_n$  are terms and  $P$  is a predicate symbol of arity  $n$ , then  $P(t_1, \dots, t_n)$  is an atomic formula
- If  $t_1$  and  $t_2$  are terms, then  $(t_1 = t_2)$  is an atomic formula
- If  $\alpha$  and  $\beta$  are formulas, and  $v$  is a variable, then  $\neg\alpha, (\alpha \wedge \beta), (\alpha \vee \beta), \exists v.\alpha, \forall v.\alpha$  are formulas

# Notation

- Occasionally add or omit  $(,)$
- Use  $[,]$  and  $\{, \}$
- Abbreviation:  $(\alpha \rightarrow \beta)$  for  $(\neg\alpha \vee \beta)$
- Abbreviation:  $(\alpha \leftrightarrow \beta)$  for  $(\alpha \rightarrow \beta) \wedge (\beta \rightarrow \alpha)$
- Predicates: mixed case capitalized, e.g., Person, OlderThan
- Functions (and constants): mixed case uncapitalized, e.g., john, father,

# Variable scope

- Free and bound occurrences of variables
- e.g.,  $P(x) \wedge \exists x[P(x) \vee Q(x)]$
- A sentence: formula with no free variables
- Substitution:  $\alpha[v/t]$  means  $\alpha$  with all free occurrences of the  $v$  replaced by term  $t$
- In general,  $\alpha[v_1/t_1, \dots, v_n/t_n]$

# Interpretations

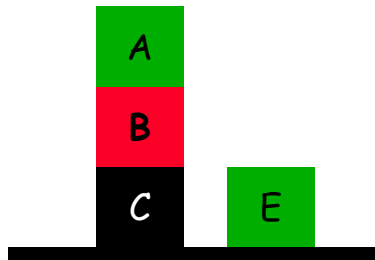
An interpretation is a pair  $\mathfrak{S} = \langle D, I \rangle$

- $D$  is the domain, can be any non-empty set
- $I$  is a mapping from the set of predicate and function symbols
- If  $P$  is a predicate symbol of arity  $n$ ,  $I(P)$  is an  $n$ -ary relation over  $D$ , i.e.,  $I(P) \subseteq D^n$ 
  - If  $p$  is a 0-ary predicate symbol, i.e., a propositional symbol,  $I(p) \in \{true, false\}$
- If  $f$  is a function symbol of arity  $n$ ,  $I(f)$  is an  $n$ -ary function over  $D$ , i.e.,  $I(f) : D^n \rightarrow D$ 
  - If  $c$  is a 0-ary function symbol, i.e., a constant symbol,  $I(c) \in D$

# Blocks world example

- $D = \{\underline{A}, \underline{B}, \underline{C}, \underline{E}\}$
- $\Phi(a) = \underline{A}, \Phi(b) = \underline{B},$   
 $\Phi(c) = \underline{C}, \Phi(e) = \underline{E}.$
- $\Psi(\text{on}) = \{(\underline{A}, \underline{B}), (\underline{B}, \underline{C})\}$
- $\Psi(\text{above}) =$   
 $\{(\underline{A}, \underline{B}), (\underline{B}, \underline{C}), (\underline{A}, \underline{C})\}$
- $\Psi(\text{clear}) = \{\underline{A}, \underline{E}\}$
- $\Psi(\text{ontable}) = \{\underline{C}, \underline{E}\}$

Environment



# Denotation of terms

- Terms denote elements of the domain
- A variable assignment  $\mu$  is a mapping from the set of variables to the domain  $D$
- $\|v\|_{\mathfrak{S}, \mu} = \mu(v)$
- $\|f(t_1, \dots, t_n)\|_{\mathfrak{S}, \mu} = I(f)(\|t_1\|_{\mathfrak{S}, \mu}, \dots, \|t_n\|_{\mathfrak{S}, \mu})$

# Satisfaction: atomic formulas

$\mathfrak{S}, \mu \models \alpha$  is read “ $\mathfrak{S}, \mu$  satisfies  $\alpha$ ”

- $\mathfrak{S}, \mu \models P(t_1, \dots, t_n)$  iff  $\langle \|t_1\|_{\mathfrak{S}, \mu}, \dots, \|t_n\|_{\mathfrak{S}, \mu} \rangle \in I(P)$
- $\mathfrak{S}, \mu \models (t_1 = t_2)$  iff  $\|t_1\|_{\mathfrak{S}, \mu} = \|t_2\|_{\mathfrak{S}, \mu}$

# Satisfaction: propositional connectives

- $\mathfrak{S}, \mu \models \neg\alpha$  iff  $\mathfrak{S}, \mu \not\models \alpha$
- $\mathfrak{S}, \mu \models (\alpha \wedge \beta)$  iff  $\mathfrak{S}, \mu \models \alpha$  and  $\mathfrak{S}, \mu \models \beta$
- $\mathfrak{S}, \mu \models (\alpha \vee \beta)$  iff  $\mathfrak{S}, \mu \models \alpha$  or  $\mathfrak{S}, \mu \models \beta$



# Satisfaction: quantifiers

$\mu\{d; v\}$  denotes a variable assignment just like  $\mu$ , except that it maps  $v$  to  $d$

- $\mathfrak{S}, \mu \models \exists v. \alpha$  iff for some  $d \in D$ ,  $\mathfrak{S}, \mu\{d; v\} \models \alpha$
- $\mathfrak{S}, \mu \models \forall v. \alpha$  iff for all  $d \in D$ ,  $\mathfrak{S}, \mu\{d; v\} \models \alpha$

Let  $\alpha$  be a sentence. Then whether  $\mathfrak{S}, \mu \models \alpha$  is independent of  $\mu$ .  
Thus we simply write  $\mathfrak{S} \models \alpha$

# Blocks world example

- $D = \{\underline{A}, \underline{B}, \underline{C}, \underline{E}\}$
- $\Phi(a) = \underline{A}, \Phi(b) = \underline{B},$   
 $\Phi(c) = \underline{C}, \Phi(e) = \underline{E}.$
- $\Psi(\text{on}) = \{(\underline{A}, \underline{B}), (\underline{B}, \underline{C})\}$
- $\Psi(\text{above}) =$   
 $\{(\underline{A}, \underline{B}), (\underline{B}, \underline{C}), (\underline{A}, \underline{C})\}$
- $\Psi(\text{clear}) = \{\underline{A}, \underline{E}\}$
- $\Psi(\text{ontable}) = \{\underline{C}, \underline{E}\}$

$\forall X, Y. \text{on}(X, Y) \rightarrow \text{above}(X, Y)$

✓  $X = \underline{A}, Y = \underline{B}$

✓  $X = \underline{C}, Y = \underline{A}$

✓ ...

$\forall X, Y. \text{above}(X, Y) \rightarrow \text{on}(X, Y)$

✓  $X = \underline{A}, Y = \underline{B}$

✗  $X = \underline{A}, Y = \underline{C}$

# Blocks world example

- $D = \{\underline{A}, \underline{B}, \underline{C}, \underline{E}\}$
- $\Phi(a) = \underline{A}, \Phi(b) = \underline{B},$   
 $\Phi(c) = \underline{C}, \Phi(e) = \underline{E}.$
- $\Psi(\text{on}) = \{(\underline{A}, \underline{B}), (\underline{B}, \underline{C})\}$
- $\Psi(\text{above}) =$   
 $\{(\underline{A}, \underline{B}), (\underline{B}, \underline{C}), (\underline{A}, \underline{C})\}$
- $\Psi(\text{clear}) = \{\underline{A}, \underline{E}\}$
- $\Psi(\text{ontable}) = \{\underline{C}, \underline{E}\}$

$\forall X \exists Y. (\text{clear}(X) \vee \text{on}(Y, X))$

✓  $X = \underline{A}$

✓  $X = \underline{C}, Y = \underline{B}$

✓ ...

$\exists Y \forall X. (\text{clear}(X) \vee \text{on}(Y, X))$

✗  $Y = \underline{A} ?$  No! ( $X = \underline{C}$ )

✗  $Y = \underline{C} ?$  No! ( $X = \underline{B}$ )

✗  $Y = \underline{E} ?$  No! ( $X = \underline{B}$ )

✗  $Y = \underline{B} ?$  No! ( $X = \underline{B}$ )

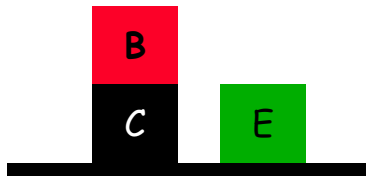
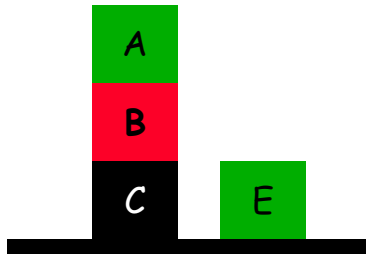
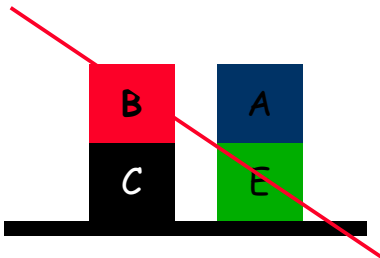
# Satisfiability

- Let  $S$  be a set of sentences
- $\mathfrak{S} \models S$ , read  $\mathfrak{S}$  satisfies  $S$ , if for every  $\alpha \in S$ ,  $\mathfrak{S} \models \alpha$
- If  $\mathfrak{S} \models S$ , we say  $\mathfrak{S}$  is a model of  $S$
- We say that  $S$  is satisfiable if there is  $\mathfrak{S}$  s.t.  $\mathfrak{S} \models S$ , and
- e.g., is  $\{\forall x(P(x) \rightarrow Q(x)), P(a), \neg Q(a)\}$  satisfiable?

# Blocks world example

KB

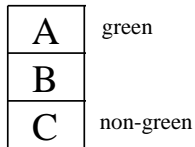
1. `on(b,c)`  
2. `clear(e)`



# Logical entailment

- $S \models \alpha$  iff for every  $\mathfrak{S}$ , if  $\mathfrak{S} \models S$  then  $\mathfrak{S} \models \alpha$
- $S \models \alpha$  is read:  $S$  entails  $\alpha$  or  $\alpha$  is a logical consequence of  $S$
- A special case:  $\emptyset \models \alpha$ , simply written  $\models \alpha$ , read “ $\alpha$  is valid”
- Note that  $\{\alpha_1, \dots, \alpha_n\} \models \alpha$  iff  $\alpha_1 \wedge \dots \wedge \alpha_n \rightarrow \alpha$  is valid iff  $\alpha_1 \wedge \dots \wedge \alpha_n \wedge \neg \alpha$  is unsatisfiable
- Alpine Club example
  - Let KB be the set of sentences, and  $\alpha$  be the question
  - We want to know if  $KB \models \alpha$ ?

# Blocks world example cont'd



- $S = \{On(a, b), On(b, c), Green(a), \neg Green(c)\}$
- $\alpha = \exists x \exists y [Green(x) \wedge \neg Green(y) \wedge On(x, y)]$
- We prove that  $S \models \alpha$

# Logical entailment: examples

- $\forall xA \vee \forall xB \models \forall x(A \vee B)$
- Does  $\forall x(A \vee B) \models \forall xA \vee \forall xB$
- $\exists x(A \wedge B) \models \exists xA \wedge \exists xB$
- Does  $\exists xA \wedge \exists xB \models \exists x(A \wedge B)$ ?
- $\exists y\forall xA \models \forall x\exists yA$
- Does  $\forall x\exists yA \models \exists y\forall xA$ ?

The only way to prove that  $KB \not\models \alpha$  is to give an interpretation satisfying KB but not  $\alpha$ .



## Alpine Club example cont'd

- Suppose that we had been told that Mike likes whatever Tony dislikes, but we had not been told that Mike dislikes whatever Tony likes.
- Can we still claim that there is a member of the Alpine Club who is a mountain climber but not a skier?
- No. We give an interpretation which satisfies the modified KB but not  $f$  as follows: Let  $D = \{T, M, J, R, S\}$ . Let  $I(tony) = T, I(mike) = M, I(john) = J, I(rain) = R, I(snow) = S$ . Let  $I(A) = \{T, M, J\}, I(S) = \{T, M, J\}, I(C) = \emptyset, I(L) = \{(T, R), (T, S), (T, T), (M, M), (M, S), (M, J), (J, S)\}$ .

# Logical entailment and knowledge-based systems

- Start with KB representing explicit beliefs, usually what the agent has been told or has learned
- Implicit beliefs:  $\{\alpha \mid KB \models \alpha\}$
- Actions depend on implicit beliefs, rather than explicit beliefs

# Inference procedure

- We want a mechanical procedure to check if  $KB \models \alpha$
- Called an inference procedure
- Sound if whenever it says yes, then  $KB \models \alpha$
- Complete if whenever  $KB \models \alpha$ , then it says yes

# Resolution-based Inference procedure

- Resolution is a rule of inference
- Resolution-based inference procedure: refutation
- We begin with the propositional case
- Then proceed to the first-order case

# Clausal form

- A literal is an atomic formula or its negation, e.g.,  $p$ ,  $\neg p$
- A clause is a disjunction of literals, written as the set of literals
  - e.g.,  $p \vee \neg r \vee s$ , written  $(p, \neg r, s)$
- A special case: empty clause  $()$ , representing false
- A formula is a conjunction of clauses, written as the set of clauses

# Resolution rule of inference

- From the two clauses  $\{p\} \cup c_1$  and  $\{\neg p\} \cup c_2$ , infer the clause  $c_1 \cup c_2$
- $c_1 \cup c_2$  is called the resolvent of input clauses wrt the atom  $p$
- e.g.,  $(p)$  and  $(\neg p)$  resolve to  $()$ ,  
 $(w, r, q)$  and  $(w, s, \neg r)$  resolve to  $(w, q, s)$  wrt  $r$
- **Proposition.**  $\{p\} \cup c_1, \{\neg p\} \cup c_2 \models c_1 \cup c_2$   
Proof:

# Derivation

A derivation of a clause  $c$  from a set  $S$  of clauses is a sequence  $c_1, c_2, \dots, c_n$  of clauses, where  $c_n = c$ , and for each  $c_i$ , either

- $c_i \in S$ , or
- $c_i$  is a resolvent of two earlier clauses in the derivation

We write  $S \vdash c$  if there is a derivation of  $c$  from  $S$

# Soundness of derivations

- **Theorem.** If  $S \vdash c$ , then  $S \models c$

Proof:

- Let  $c_1, c_2, \dots, c_n$  be a derivation of  $c$  from  $S$
  - We prove by induction on  $i$  that for all  $1 \leq i \leq n$ ,  $S \models c_i$ .
- However, the converse does not hold in general  
e.g.,  $(p) \models (p, q)$ , but  $(p) \not\vdash (p, q)$



# Soundness and completeness of refutations

**Theorem.**  $S \vdash ()$  iff  $S \models ()$  iff  $S$  is unsatisfiable

We will not prove the completeness part

# Resolution-based inference procedure: refutation

$KB \models \alpha$  iff  $KB \wedge \neg\alpha$  is unsatisfiable

Thus to check if  $KB \models \alpha$ ,

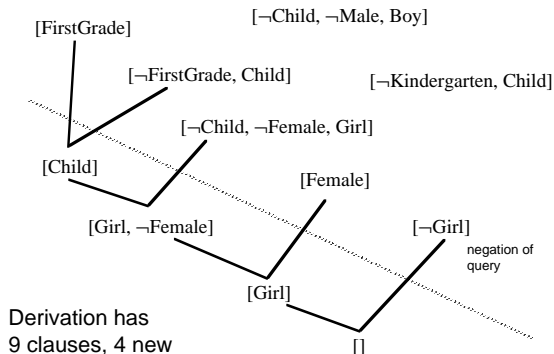
- put KB and  $\neg\alpha$  into clausal form to get  $S$ ,
- check if  $S \vdash ()$

# Refutation example 1

KB

FirstGrade  
FirstGrade  $\supset$  Child  
Child  $\wedge$  Male  $\supset$  Boy  
Kindergarten  $\supset$  Child  
Child  $\wedge$  Female  $\supset$  Girl  
Female

Show that  $\text{KB} \models \text{Girl}$

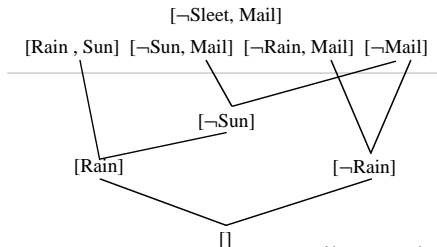


# Refutation example 2

KB

$(\text{Rain} \vee \text{Sun})$   
 $(\text{Sun} \supset \text{Mail})$   
 $((\text{Rain} \vee \text{Sleet}) \supset \text{Mail})$

Show  $\text{KB} \models \text{Mail}$



Note: every clause  
not in  $S$  has 2 parents

Similarly  $\text{KB} \not\models \text{Rain}$

Can enumerate all resolvents given  $\neg\text{Rain}$ ,  
and  $[\ ]$  will not be generated

# The first-order case

We need

- A way of converting KB and  $f$  (the query) into clausal form
- A way of doing resolution even when we have variables.  
This needs unification

# Conversion to Clausal Form

- 1 Eliminate Implications.
- 2 Move Negations inwards (and simplify  $\neg\neg$ ).
- 3 Standardize Variables.
- 4 Skolemize.
- 5 Convert to Prenex Form.
- 6 Distribute disjunctions over conjunctions.
- 7 Flatten nested conjunctions and disjunctions.
- 8 Convert to Clauses.

Consider  $\exists y. Elephant(y) \wedge Friendly(y)$

- This asserts that there is some individual that is both an elephant and friendly.
- To remove the existential, we invent a name for this individual, say  $a$ . This is a new constant symbol not equal to any previous constant symbols:  
 $Elephant(a) \wedge Friendly(a)$
- This is saying the same thing, since we do not know anything about the new constant  $a$ .
- It is essential that the introduced symbol  $a$  is new. Else we might say more than the existential formula.

# Skolemization

Now consider  $\forall x \exists y. \text{Loves}(x, y)$ .

- This formula claims that for every  $x$  there is some  $y$  that  $x$  loves (perhaps a different  $y$  for each  $x$ ).
- Replacing the existential by a new constant won't work:  $\forall x. \text{Loves}(x, a)$ , because this asserts that there is a particular individual  $a$  loved by every  $x$ .
- To properly convert existential quantifiers scoped by universal quantifiers we must use functions not just constants.
- In this case  $x$  scopes  $y$ , so we must replace  $y$  by a function of  $x$ :  $\forall x. \text{Loves}(x, g(x))$ , where  $g$  is a new function symbol.
- This formula asserts that for every  $x$  there is some individual (given by  $g(x)$ ) that  $x$  loves.  $g(x)$  can be different for each  $x$ .



# Skolemization examples

- $\forall x, y, z \exists w. R(x, y, z, w) \implies \forall x, y, z. R(x, y, z, h_1(x, y, z))$
- $\forall x, y \exists w. R(x, y, g(w)) \implies \forall x, y. R(x, y, g(h_2(x, y)))$
- $\forall x, y \exists w \forall z. R(x, y, w) \wedge Q(z, w) \implies$   
 $\forall x, y, z. R(x, y, h_3(x, y)) \wedge Q(z, h_3(x, y))$

# A conversion example

$$\forall x \{ P(x) \rightarrow [\forall y (P(y) \rightarrow P(f(x, y))) \wedge \neg \forall y (\neg Q(x, y) \wedge P(y))] \}$$

1. Eliminate implications using  $A \rightarrow B \Leftrightarrow \neg A \vee B$

$$\forall x \{ \neg P(x) \vee [\forall y (\neg P(y) \vee P(f(x, y))) \wedge \neg \forall y (\neg Q(x, y) \wedge P(y))] \}$$

# Move negations inwards

$$\forall x \{ \neg P(x) \vee [\forall y (\neg P(y) \vee P(f(x, y))) \wedge \neg \forall y (\neg Q(x, y) \wedge P(y))] \}$$

2. Move negations inwards using

- $\neg(A \vee B) \Leftrightarrow \neg A \wedge \neg B$ ,  $\neg(A \wedge B) \Leftrightarrow \neg A \vee \neg B$
- $\neg \exists x.A \Leftrightarrow \forall x.\neg A$ ,  $\neg \forall x.A \Leftrightarrow \exists x.\neg A$ ,  $\neg \neg A \Leftrightarrow A$

$$\forall x \{ \neg P(x) \vee [\forall y (\neg P(y) \vee P(f(x, y))) \wedge \exists y (Q(x, y) \vee \neg P(y))] \}$$

# Standardize Variables

$$\forall x \{ \neg P(x) \vee [\forall y (\neg P(y) \vee P(f(x, y))) \wedge \exists y (Q(x, y) \vee \neg P(y))] \}$$

3. Standardize Variables (Rename variables so that each quantified variable is unique)

$$\forall x \{ \neg P(x) \vee [\forall y (\neg P(y) \vee P(f(x, y))) \wedge \exists z (Q(x, z) \vee \neg P(z))] \}$$

$$\forall x \{ \neg P(x) \vee [\forall y (\neg P(y) \vee P(f(x, y))) \wedge \exists z (Q(x, z) \vee \neg P(z))] \}$$

4. Skolemize (Remove existential quantifiers by introducing new function symbols)

$$\forall x \{ \neg P(x) \vee [\forall y (\neg P(y) \vee P(f(x, y))) \wedge (Q(x, g(x)) \vee \neg P(g(x)))] \}$$

# Convert to prenex form

$$\forall x \{ \neg P(x) \vee [\forall y (\neg P(y) \vee P(f(x, y))) \wedge (Q(x, g(x)) \vee \neg P(g(x)))] \}$$

5. Convert to prenex form. (Bring all quantifiers to the front – only universals, each with different name)

$$\forall x \forall y \{ \neg P(x) \vee [(\neg P(y) \vee P(f(x, y))) \wedge (Q(x, g(x)) \vee \neg P(g(x)))] \}$$

# Disjunctions over conjunctions

$$\forall x \forall y \{ \neg P(x) \vee [(\neg P(y) \vee P(f(x, y))) \wedge (Q(x, g(x)) \vee \neg P(g(x)))] \}$$

6. Disjunctions over conjunctions using

$$A \vee (B \wedge C) \Leftrightarrow (A \vee B) \wedge (A \vee C)$$

$$\forall x \forall y \{ (\neg P(x) \vee \neg P(y) \vee P(f(x, y))) \wedge \\ (\neg P(x) \vee Q(x, g(x)) \vee \neg P(g(x))) \}$$

# Convert to Clauses

$$\forall x \forall y \{ (\neg P(x) \vee \neg P(y) \vee P(f(x, y))) \wedge \\ (\neg P(x) \vee Q(x, g(x)) \vee \neg P(g(x))) \}$$

8. Convert to Clauses (remove quantifiers and break apart conjunctions).

a)  $\neg P(x) \vee \neg P(y) \vee P(f(x, y))$

b)  $\neg P(x) \vee Q(x, g(x)) \vee \neg P(g(x))$



# Unification

- Can the clauses  $(P(john), Q(fred), R(x))$  and  $(\neg P(y), R(susan), R(y))$  be resolved?
- Once reduced to clausal form, all remaining variables are universally quantified.
- So, implicitly the clause  $(P(john), Q(fred), R(x))$  represents  $(P(john), Q(fred), R(john))$ ,  $(P(john), Q(fred), R(fred))$ , ...
- So there is a specialization of  $(P(john), Q(fred), R(x))$  that can be resolved with a specialization of  $(\neg P(y), R(susan), R(y))$
- In particular,  $(P(john), Q(fred), R(john))$  can be resolved with  $(\neg P(john), R(susan), R(john))$ , producing  $(Q(fred), R(john), R(susan))$

# Unification

- We want to be able to match conflicting literals, even when they have variables.
- This matching process automatically determines whether or not there is a specialization that matches.
- But, we don't want to over specialize!

# Unification

- Consider  $(\neg P(x), S(x), Q(fred))$  and  $(P(y), R(y))$
- We need to unify  $P(x)$  and  $P(y)$ . How do we do this?
- Possible resolvents:
  - $(S(john), Q(fred), R(john))\{x = john, y = john\}$
  - $(S(sally), Q(fred), R(sally))\{x = sally, y = sally\}$
  - $(S(x), Q(fred), R(x))\{y = x\}$
- The last resolvent is most-general, the other two are specializations. We want the most general clause for use in future resolution steps.

# Substitution

- Unification is a mechanism for finding a most general matching
- A key component of unification is substitution.
- A substitution is a finite set of equations of the form  $V = t$  where  $V$  is a variable and  $t$  is a term not containing  $V$ . ( $t$  might contain other variables).
- We can apply a substitution  $\sigma = \{V_1 = t_1, \dots, V_n = t_n\}$  to a formula  $f$  to obtain a new formula  $f\sigma$  by simultaneously replacing every variable  $V_i$  by term  $t_i$ .
- e.g.,  $P(x, g(y, z))\{x = y, y = f(a)\} \implies P(y, g(f(a), z))$
- Note that the substitutions are not applied sequentially, i.e., the first  $y$  is not subsequently replaced by  $f(a)$ .

# Composition of substitutions

- We can compose two substitutions  $\theta$  and  $\sigma$  to obtain a new substitution  $\theta\sigma$
- Let  $\theta = \{x_1 = s_1, x_2 = s_2, \dots, x_m = s_m\}$ ,  
 $\sigma = \{y_1 = t_1, y_2 = t_2, \dots, y_k = t_k\}$
- Step 1. Get  $S = \{x_1 = s_1\sigma, x_2 = s_2\sigma, \dots, x_m = s_m\sigma,$   
 $y_1 = t_1, y_2 = t_2, \dots, y_k = t_k\}$
- Step 2. Delete any identities, *i.e.*, equations of the form  $V = V$ .
- Step 3. Delete any equation  $y_i = s_i$  where  $y_i$  is equal to one of the  $x_j$  in  $\theta$ . Why?

# Composition example

- Let  $\theta = \{x = f(y), y = z\}$ ,  $\sigma = \{x = a, y = b, z = y\}$
- Step 1. Get  $S = \{x = f(b), y = y, x = a, y = b, z = y\}$
- Step 2. Delete  $y = y$ .
- Step 3. Delete  $x = a$ .
- The result is  $S = \{x = f(b), y = b, z = y\}$

# Note on substitutions

- The empty substitution  $\epsilon = \{\}$  is also a substitution, and we have  $\theta\epsilon = \theta$ .
- More importantly, substitutions when applied to formulas are associative:  $(f\theta)\sigma = f(\theta\sigma)$
- Composition is simply a way of converting the sequential application of a series of substitutions to a single substitution.

- A unifier of two formulas  $f$  and  $g$  is a substitution  $\sigma$  that makes  $f$  and  $g$  syntactically identical.
- Note that not all formulas can be unified – substitutions only affect variables.
- e.g.,  $P(f(x), a)$  and  $P(y, f(w))$  cannot be unified, as there is no way of making  $a = f(w)$  with a substitution.



A substitution  $\sigma$  of two formulas  $f$  and  $g$  is a Most General Unifier (MGU) if

- $\sigma$  is a unifier.
- For every other unifier  $\theta$  of  $f$  and  $g$  there must exist a third substitution  $\lambda$  such that  $\theta = \sigma\lambda$ .

This says that every other unifier is “more specialized” than  $\sigma$ .

The MGU of a pair of formulas  $f$  and  $g$  is unique up to renaming.

# MGU example

- $P(f(x), z)$  and  $P(y, a)$
- $\sigma = \{y = f(a), x = a, z = a\}$  is a unifier, but not an MGU
- $\theta = \{y = f(x), z = a\}$  is an MGU
- $\sigma = \theta\lambda$ , where  $\lambda = \{x = a\}$

# Computing MGUs

- The MGU is the “least specialized” way of making atomic formulas with variables match.
- We can compute MGUs.
- Intuitively we line up the two formulas and find the first sub-expression where they disagree.
- The pair of subexpressions where they first disagree is called the disagreement set.
- The algorithm works by successively fixing disagreement sets until the two formulas become syntactically identical.

# Computing MGUs

Given two atomic formulas  $f$  and  $g$

- ①  $k = 0$ ;  $\sigma_0 = \{\}$ ;  $S_0 = \{f, g\}$
- ② If  $S_k$  contains an identical pair of formulas, stop and return  $\sigma_k$  as the MGU of  $f$  and  $g$ .
- ③ Else find the disagreement set  $D_k = \{e_1, e_2\}$  of  $S_k$
- ④ If  $e_1 = V$  a variable, and  $e_2 = t$  a term not containing  $V$  (or vice-versa) then let  $\sigma_{k+1} = \sigma_k\{V = t\}$ ;  $S_{k+1} = S_k\{V = t\}$ ;  $k = k + 1$ ; Goto 2
- ⑤ Else stop,  $f$  and  $g$  cannot be unified.

# Computing MGU examples

- ①  $P(f(a), g(x))$  and  $P(y, y)$
- ②  $P(a, x, h(g(z)))$  and  $P(z, h(y), h(y))$
- ③  $P(x, x)$  and  $P(y, f(y))$

# First-order Resolution

From the two clauses  $\{\rho_1\} \cup c_1$  and  $\{\neg\rho_2\} \cup c_2$ , where there exists a MGU  $\sigma$  for  $\rho_1$  and  $\rho_2$ , infer the clause  $(c_1 \cup c_2)\sigma$

**Theorem.**  $S \vdash ()$  iff  $S$  is unsatisfiable

# A resolution example

1.  $(P(x), Q(g(x)))$
2.  $(R(a), Q(z), \neg P(a))$
3.  $R[1a, 2c]\{X=a\} (Q(g(a)), R(a), Q(z))$

- “R” means resolution step.
- “1a” means the 1st (a-th) literal in the first clause:  $P(x)$ .
- “2c” means the 3rd (c-th) literal in the second clause:  $\neg P(a)$ .
- 1a and 2c are the “clashing” literals.
- $\{X = a\}$  is the MGU applied.

# Refutation example 1

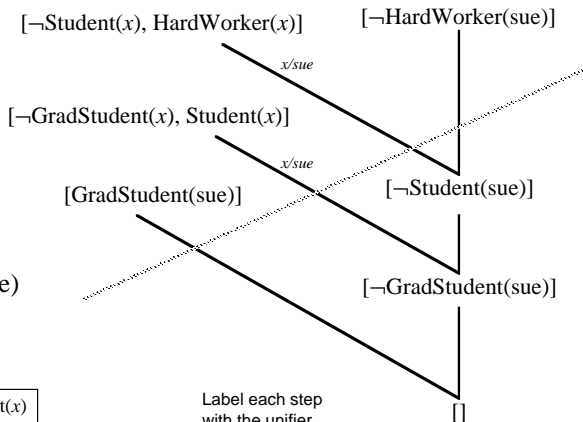
?  
KB  $\models$  HardWorker(sue)

KB

$\forall x \text{ GradStudent}(x) \supset \text{Student}(x)$

$\forall x \text{ Student}(x) \supset \text{HardWorker}(x)$

$\text{GradStudent}(\text{sue})$



Label each step  
with the unifier

Point to relevant  
literals in clauses



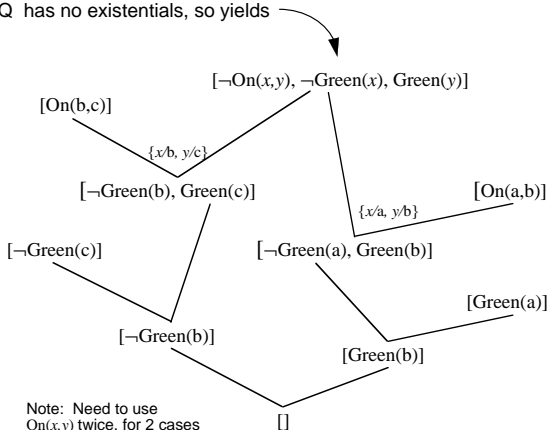
# The 3 blocks example

KB = {On(a,b), On(b,c), Green(a),  $\neg$ Green(c)}

already in CNF

Query =  $\exists x \exists y [\text{On}(x,y) \wedge \text{Green}(x) \wedge \neg \text{Green}(y)]$

Note:  $\neg Q$  has no existentials, so yields



Note: Need to use  
 $\text{On}(x,y)$  twice, for 2 cases

# Alpine Club example

$\forall x(A(x) \wedge \neg S(x)) \rightarrow C(x)$	$\Rightarrow$	1. $A(tony)$
$\forall x(C(x) \rightarrow \neg L(x, rain))$	$\Rightarrow$	2. $A(mike)$
$\forall x(\neg L(x, snow) \rightarrow \neg S(x))$	$\Rightarrow$	3. $A(john)$
$\forall x(L(tony, x) \rightarrow \neg L(mike, x))$	$\Rightarrow$	4. $L(tony, rain)$
$\forall x(\neg L(tony, x) \rightarrow L(mike, x))$	$\Rightarrow$	5. $L(tony, snow)$
$\neg \exists x(A(x) \wedge C(x) \wedge \neg S(x))$	$\Rightarrow$	6. $(\neg A(x), S(x), C(x))$
		7. $(\neg C(y), \neg L(y, rain))$
		8. $(L(z, snow), \neg S(z))$
		9. $(\neg L(tony, u), \neg L(mike, u))$
		10. $(L(tony, v), L(mike, v))$
		11. $(\neg A(w), \neg C(w), S(w))$

Note that we must standardize variables.

# Alpine Club example refutation

- 12.  $R[5, 9a] u = \text{snow} \neg L(\text{mike}, \text{snow})$
- 13.  $R[8, 12] z = \text{mike} \neg S(\text{mike})$
- 14.  $R[6b, 13] x = \text{mike} (\neg A(\text{mike}), C(\text{mike}))$
- 15.  $R[2, 14a] C(\text{mike})$
- 16.  $R[8a, 12] z = \text{mike} \neg S(\text{mike})$
- 17.  $R[2, 11] w = \text{mike} (\neg C(\text{mike}), S(\text{mike}))$
- 18.  $R[15, 17] S(\text{mike})$
- 19.  $R[16, 18] ()$

# Refutation examples

Prove that  $\exists y \forall x P(x, y) \models \forall x \exists y P(x, y)$

- $\exists y \forall x P(x, y) \Rightarrow 1. P(x, a)$
- $\neg \forall x \exists y P(x, y) \Leftrightarrow \exists x \forall y \neg P(x, y) \Rightarrow 2. \neg P(b, y)$
- $R[1,2]\{x = b, y = a\}()$

Exercises: Prove

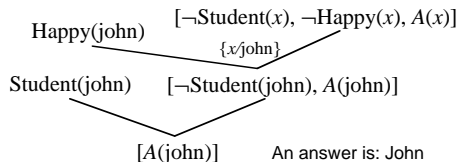
- $\forall x P(x) \vee \forall x Q(x) \models \forall x (P(x) \vee Q(x))$
- $\exists x (P(x) \wedge Q(x)) \models \exists x P(x) \wedge \exists x Q(x)$

# Answer extraction

- We can also answer wh- questions
- Replace query  $\exists x P(x)$  by  $\exists x [P(x) \wedge \neg \text{answer}(x)]$
- Instead of deriving  $()$ , derive any clause containing just the answer predicate

KB: Student(john)  
Student(jane)  
Happy(john)

Q:  $\exists x [\text{Student}(x) \wedge \text{Happy}(x)]$



# Alpine Club example answer extraction

- 11.  $(\neg A(w), \neg C(w), S(w), \text{answer}(w))$
- The same resolution steps as before give us  $\text{answer}(\text{mike})$

# Disjunctive answers

KB:

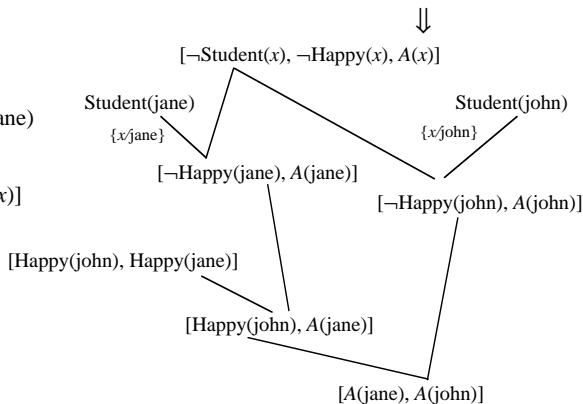
Student(john)

Student(jane)

Happy(john)  $\vee$  Happy(jane)

Query:

$\exists x[\text{Student}(x) \wedge \text{Happy}(x)]$



An answer is: either Jane or John

Note:

# A problem

KB:

$\text{LessThan}(\text{succ}(x), y) \supset \text{LessThan}(x, y)$

Query:

$\text{LessThan}(\text{zero}, \text{zero})$

Should fail since  $\text{KB} \not\models Q$

$[\text{LessThan}(x, y), \neg \text{LessThan}(\text{succ}(x), y)]$

$[\neg \text{LessThan}(0, 0)]$

$x/0, y/0$

$[\neg \text{LessThan}(1, 0)]$

$x/1, y/0$

$[\neg \text{LessThan}(2, 0)]$

$x/2, y/0$

...

...

Infinite branch of resolvents

We use 0 for zero, 1 for  $\text{succ}(\text{zero})$ , 2 for  $\text{succ}(\text{succ}(\text{zero}))$ , ...



# Undecidability in the first-order case

- There can be no procedure to decide if a set of clauses is satisfiable.
- **Theorem.**  $S \vdash ()$  iff  $S$  is unsatisfiable
- However, there is no procedure to check if  $S \vdash ()$ , because
- When  $S$  is satisfiable, the search for  $()$  may not terminate

# Intractability in the propositional case

- Determining if a set of clauses is satisfiable was shown by Cook in 1972 to be NP-complete.
- Satisfiability is believed by most people to be unsolvable in polynomial time.
- Procedures have been proposed for determining satisfiability that appear to work much better in practice than Resolution.
- They are called SAT solvers as they are mostly used to find a satisfying interpretation for clauses that are satisfiable.

# Implications for KRR

- In knowledge-based systems, actions depend on implicit beliefs, *i.e.*, logical entailments of KB
- However, as we have seen, computing entailments is unsolvable in general
- The hope is that in many practical scenarios, entailments can be efficiently computed
- In case entailments are difficult to compute, we seek for other ways out

# Prolog and resolution

- Resolution forms the basis of the implementation of Prolog
- When searching for  $()$ , Prolog uses a specific depth-first left-right strategy

# Refutation exercise

- Some patients like all doctors.
- No patient likes any quack.
- Therefore no doctor is a quack.

Use predicates:  $P(x)$ ,  $D(x)$ ,  $Q(x)$ ,  $L(x, y)$

# Refutation exercise

- Whoever can read is literate.
- Dolphins are not literate.
- Flipper is an intelligent dolphin.
- Who is intelligent but cannot read.

Use predicates:  $R(x)$ ,  $L(x)$ ,  $D(x)$ ,  $I(x)$