Knowledge Representation Predicate Calculus & Automated Reasoning for AI

Source s

1/ George Luger, Artificial Intelligence: Structures and Strategies for Complex Problem, Addison-Wesley. chap. 2 & section 14.2

2/ Stuart Russell and Peter Norvig, Artificial Intelligence: A Modern Approach, Prentice Hall. sections 7.1-7.5, chap 8, chap 9

3/Some slides and/or pictures in the following are adapted from slides ©2012. Prof. L. Kosseim;

-

The Physical Symbol System

- also known as GOFAI (*Good Old Fashion AI*)
- Newel & Simon argue that intelligent activity is achieved by the use of:
  1. Symbol patterns to represent the problem domain
  2. Operations on these patterns to generate potential solutions
  3. Search to select a solution from among these possibilities

- Alternative approaches to the symbol system:
  - neural networks, genetic algorithms, statistical approaches…

Agenda

- Propositional Logics
- Predicate Logics
- Automated reasoning
  - Deduction
    - Forward Chaining
    - Backward Chaining
  - Resolution-Refutation

Logic

- One way to represent facts and reason with them
  - i.e. to derive new knowledge from old (through inference)
  - we can prove that a new statement is true by proving that it follows from the statements that we already know

- Three uses of logic in AI:
  1. for knowledge representation
  2. for reasoning
  3. as a programming language - Prolog

Logic can be regarded as a formalization of the language of human thoughts

- Used in early domains of AI:
  - ex: mechanical theorem proving
    - *Logic Theorist* by Newell et al. 1963, proved most theorems of chap. 2 of *Principia Mathematica*
    - *General Problem Solver*

Why Use Logics?

- Why not plain natural language?
- Problem 1: NL is too ambiguous
  - *Every man loves a woman.*
  - *The man saw the boy with the telescope.*
  - *John, Mary, my cousin, and Jim are all late.*
- Depends on the meaning/semantics of the sentences
- We need a formal language free of ambiguity

Why Use Logics?

- Problem 2: Need a reasoning mechanism
  - E.g.. : Typical *River crossing* problem

    Problem: *A* man has a wolf, a goat, and a cabbage. He must cross a river with the two animals and the cabbage. There is a small rowing-boat, in which he can take only one thing with him at a time. If, however, the wolf and the goat are left alone, the wolf will eat the goat. If the goat and the cabbage are left alone, the goat will eat the cabbage.

    Question: How can the man get across the river with the two animals and the cabbage?

    Have fun!

Many Logic Formalisms

- Propositional logic
- Predicate logic

} The focus of this class

- Fuzzy logic – for reasoning under uncertainty
- Default logics
- Non-monotonic logic
- Description logics
- Non-standard logics
- …

Later in this course

Main Logic Formalisms

- Propositional logic
  - Well-defined **formal semantics**
  - **Sound** and **complete** inference rules
  - But expressive power is too restrictive

- Predicate logic
  - Well-defined **formal semantics**
  - **Sound** and **complete** inference rules
  - But more expressive power

Today

- Propositional Logics
- Predicate Logics
- Automated reasoning
  - Deduction
    - Forward Chaining
    - Backward Chaining
  - Resolution-Refutation

Propositional Logic: Syntax

- Defines the symbols:
  - Truth symbols: True False
  - Propositional symbols: P, Q, …
  - Connectors: $\neg$, $\wedge$, $\vee$, $\Rightarrow$ (or $\rightarrow$), $\Leftrightarrow$ (or $\equiv$) *in decreasing precedence*
- Defines what is a sentence:
  - Truth symbols and proposition symbols are sentences
    - E.g.. True P Q
  - If P is a sentence, then $\neg$P and (P) are sentences
  - If P and Q are sentences, then P$\wedge$Q, P$\vee$Q, P$\rightarrow$Q, P$\Leftrightarrow$Q are sentences
  - legal sentences are also called WWFs (Well-Formed Formulas)

Example

- Consider the example
  - If Socrates is human, then Socrates is mortal
  - If Socrates is Greek, then Socrates is human
  - Socrates is Greek
  - Therefore, Socrates is mortal
- Propositions are
  - "Socrates is human" = P
  - "Socrates is mortal" = Q
  - "Socrates is Greek" = R
- After substituting symbols, we get
  - $P \rightarrow Q$
  - $R \rightarrow P$
  - R
  - $((P \rightarrow Q) \wedge (R \rightarrow P) \wedge R) \rightarrow Q$

Propositional Logic: Semantics

- Semantics: defines the meaning of a sentence (its truth value)

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$	$P \Leftrightarrow Q$
True	True	False	True	True	True	True
True	False	False	False	True	False	False
False	True	True	False	True	True	False
False	False	True	False	False	True	True

Some Laws on Propositional Logics

- Double-negation elimination:
  - $\neg(\neg P) \equiv P$
- $(P \vee Q) \equiv (\neg P \Rightarrow Q)$
- Contrapositive law:
  - $(P \Rightarrow Q) \equiv (\neg Q \Rightarrow \neg P)$
- de Morgan's law:
  - $\neg(P \wedge Q) \equiv \neg P \vee \neg Q$
  - $\neg(P \vee Q) \equiv \neg P \wedge \neg Q$

Some Laws on Propositional Logics

- Commutative law:
  - $(P \wedge Q) \equiv (Q \wedge P)$
  - $(P \vee Q) \equiv (Q \vee P)$
- Associative law:
  - $((P \wedge Q) \wedge R) \equiv (P \wedge (Q \wedge R))$
  - $((P \vee Q) \vee R) \equiv (P \vee (Q \vee R))$
- Distributive law:
  - $P \vee (Q \wedge R) \equiv (P \vee Q) \wedge (P \vee R)$
  - $P \wedge (Q \vee R) \equiv (P \wedge Q) \vee (P \wedge R)$

From NL to Propositional Logic

Rains	R	*Pic-nic cancelled*	C
Be wet	W	*Stay at home*	H

- *If it rains and I stay home, then I will not be wet.*

$(R \wedge H) \rightarrow \neg W$

- *I will be wet if it rains.*

$R \rightarrow W$

- *If it rains and the pic-nic is not cancelled or I do not stay at home, I will be wet.*

$(R \wedge (\neg C \vee \neg H)) \rightarrow W$ or $(R \wedge \neg C) \vee \neg H) \rightarrow W$

- *Whether the pic-nic is cancelled or not, I am staying home if it rains.*

$(C \vee \neg C) \wedge R \rightarrow H$

Problem with Propositional Logic…

- How do you represent:
  - *It rained Wednesday*
  - *It rained Friday*
  - *It rained Saturday*
  - …
- We'd like to access components of the sentence
  - e.g. raining & the day of the week

- Lack of expressive power
- Need predicate logic

Today

- Propositional Logics
- Predicate Logics
- Automated reasoning
  - Deduction
    - Forward-chaining
    - Backward-chaining
  - Resolution-Refutation

Predicate Calculus

- *It rained Sunday* -->
  - `weather(sunday, rain)`
- *Bill jogged on Sunday* -->
  - `jogged(bill, sunday)` or
  - `activity(bill, jogging, sunday)`
  - or…
- Now we can formally find out that `sunday` is mentioned in both formulas and do more advanced reasoning

First-Order Predicate Logic (FOPL): Syntax

- Symbols:
  - letters, digits, _
  - begin with a letter
  - used to denote objects, properties or relations in the world
  - + numbers & arithmetic operations

- Symbols:
  1. Truth symbols ex: true, false
  2. Constants ex: `a, b, mary`
  3. Variables ex: `X, Y, …`
  4. Functions ex: `leftHandOf(X), fatherOf(X), …`
  5. Predicates ex: `leftHandOf(X,Y), fatherOf(X,Y), …`

FOPL:

- Terms:
  - used to denote objects & properties (can be arguments)
  - Truth symbol, Constants, Functions, Variables
    - ex. `mary, wifeOf(john), X, price(apples)`
- Predicates
  - used to denote relationships between terms
    - ex: `brother, human, female`
- Sentence:
  - Atomic sentence
    - a predicate with its arguments
    - ex. `brother(john,jim)`
  - Non-atomic sentence
    - if S and T are sentences, then
      - $\neg S$, $S \wedge T$, $S \vee T$, $S \rightarrow T$, $S \Leftrightarrow T$ are sentences
      - $\exists X\ S$ is a sentence
      - $\forall X\ S$ is a sentence

Symbols May Represent...

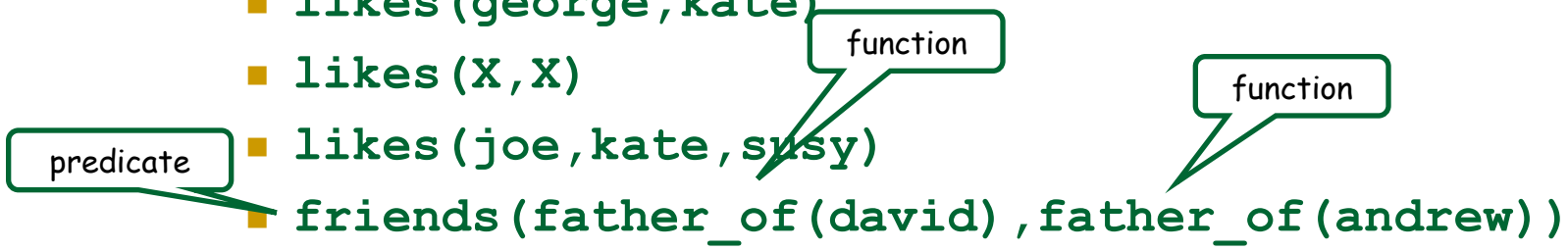
- Constants (e.g. bill, rain, sunday)
  - name specific objects or properties in the world
  - always begin with a lowercase letter
  - true and false are reserved as truth symbols
- Variables (e.g.. X, Y)
  - denote general classes of objects or properties in the world
  - allow abstraction and generalization
  - always begin with an uppercase letter

Symbols may represent...

- Functions (e.g.. `father`, `plus`)
  - begin with a lowercase letter
  - map members of a set to a unique member of a set
  - i.e. evaluate to a value
    - `father(david)` evaluates to `george`
    - `plus(2,3)` evaluates to `5`
  - has an arity (nb of arguments)
    - ex: `father/1`
    - note: `father/1` is different from `father/2`
  - form function expressions together with their arguments
    - `f(X,Y)`, `father(david)`, `price(bananas)`, `plus(2,3)`

Symbols May Represent…

- Predicates
  - begin with a lowercase letter
  - name relationships between objects in the world
  - i.e. evaluate to T or F
    - `father(david,george)` evaluates to T
  - has an arity (nb of arguments)
    - ex: `father/1`
    - note: `father/1` is different from `father/2`
  - ex:
    - `likes(george,kate)`
    - `likes(X,X)`
    - `likes(joe,kate,susy)`
    - `friends(father_of(david),father_of(andrew))`



Quantifiers

- universal quantification: $\forall$
  - $\forall X$ weather(X,rain)
  - $\forall X$ likes(X,ice_cream)
- existential quantification: $\exists$
  - $\exists X$ weather(X,rain)
  - $\exists Y$ friends(Y,peter)
- Quantifiers can be arbitrarily nested
  - $\forall X \exists Y$ human(X) $\wedge$ parent(X,Y) $\wedge$ human(Y)
  - $\forall X \exists Y \forall Z$ p(X,Y) $\wedge$ q(Y,Z)

Examples

- *Floyd is a cat* --> cat(floyd)
- If *Floyd is a cat, then it is smart* --> cat(floyd) $\rightarrow$ smart(floyd)
- cat(floyd) $\vee$ $\neg$cat(floyd) is a tautology
- $\forall$X (cat(X) $\vee$ dog(X)) $\wedge$ lives_in(X,park) $\rightarrow$ happy(X)
- $\forall$X $\forall$Y $\forall$Z abbreviated as $\forall$X,Y,Z
- $\forall$X,Y,Z equal(X,Y) $\wedge$ equal(Y,Z) $\rightarrow$ equal(X,Z)

More Examples

1. *Marcus was a man.*
   man(marcus)
2. *Marcus was from Pompae.*
   fromPompae(marcus)
3. *All people from Pompae were romans.*
   $\forall X$ fromPompae(X) $\rightarrow$ roman(X)
4. *Caesar was a leader.*
   leader(caesar)
5. *All romans were loyal to Caesar or hated him.*
   $\forall X$ roman(X) $\rightarrow$ (loyalTo(X,caesar) $\vee$ hated(X,caesar))
6. *Everyone is loyal to someone.*
   $\forall X \exists Y$ loyalTo(X,Y)
7. *People only try to assassinate leaders to whom they are not loyal.*
   $\forall X \forall Y$ person(X)$\wedge$leader(Y)$\wedge$tryToAssasinate(X,Y) $\rightarrow \neg$loyalTo(X,Y)
8. *Marcus tried to assassinate Caesar*
   tryToAssasinate(marcus,caesar)

Trivia on Quantifiers

true or false?

- $\forall X\ \forall Y$ pred(X,Y) is the same as $\forall Y\ \forall X$ pred(X,Y) ???
- $\exists X\ \exists Y$ pred(X,Y) is the same as $\exists Y\ \exists X$ pred(X,Y) ???

- $\exists X\ \forall Y$ pred(X,Y) is the same as $\forall Y\ \exists X$ pred(X,Y) ???

$\exists X\ \forall Y$ loves(Y,X)

There exists a person X that everybody loves

$\forall Y\ \exists X$ loves(Y,X)

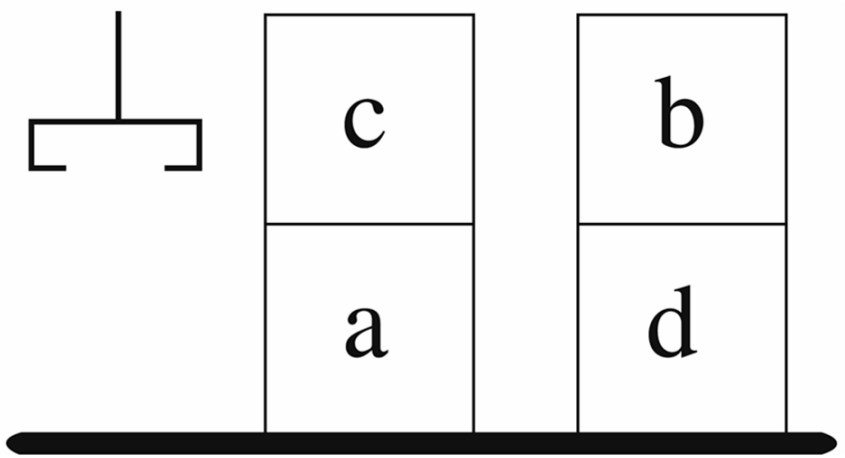
Everybody loves a person X (not necessary the same X)

Why is it Called First-Order Predicate Calculus?

- because …
- E.g:
  - ∃X loves(john,X) --> OK
  - ∃X X(john,mary) --> not OK

- you need higher-order logic to do that.

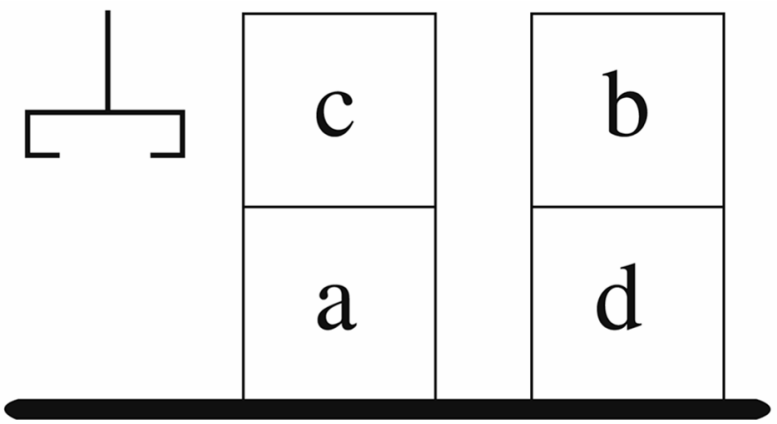
Example: The Blocks World



- Represent the Blocks World with predicate logics
- Useful to control a robot arm, for ex.
- Predicates represent qualitative relationships
  - does a block have a clear top surface?
  - can we pick up a block?

Description of the World

- Description of a snapshot of blocks world
  - on(c,a)
  - on(b,d)
  - ontable(a)
  - ontable(d)
  - clear(b)
  - clear(c)
  - hand_empty



Inference Rules

- When is a block clear?

  $\forall X\ (\neg\exists Y\ on(Y,X) \rightarrow clear(X))$

- How do we stack a clear block X on top of a clear block Y?

  $\forall X\ \forall Y\ ((\quad hand\_empty$
  $\wedge\ clear(X)$
  $\wedge\ clear(Y)$
  $\wedge\ pick\_up(X)$
  $\wedge\ put\_down(X,Y))$
  $\rightarrow\ stack(X,Y))$

- now we can reason to go from one block configuration to another

Substitution/Unification

- to reason with first-order logic we need to match:
  - sentences that contain variables against
  - other sentences that may or may not contain variables.

  E.g:
  - p(X) matches p(jack) with X=jack
  - q(father_of(X), Y) matches q(Y,Z)
    - with Y=father_of(X) and Z=Y
    - the result of the match is q(father_of(X),father_of(X))

- Substitutions are a way to keep track of the value (or binding) of variables during matching
- Notation: $\sigma = \{new_1/old_1, new_2/old_2, \ldots, new_n/old_n\}$

Examples of Substitution

$A = \forall X\ cat(X) \Rightarrow annoying(X)$

$\sigma = \{felix/X\}$

$A\sigma = cat(felix) \Rightarrow annoying(felix)$

$A = \forall X\ \forall Y\ near(X,Y) \Rightarrow near(Y,X)$

$\sigma = \{john/X, mary/Y\}$

$A\sigma = near(john,mary) \Rightarrow near(mary,john)$

Unification

- if a substitution can be applied to a set of clauses A and B such that $A\sigma = B\sigma$,
- then σ is a unifier for A and B

A= cat(X)
B= cat(felix)
σ= {felix/X}

A= cat(X)
B= cat(Y)
σ= {Y/X} or σ= {felix/X, felix/Y}

A= friend(john,X)
B= friend(Y, Z)
σ= {john/Y, Z/X} or
σ= {john/Y, mary/X, mary/Z} or
σ= {john/Y, ali/X, ali/Z} or …

The unifier does not always exist...

- Let P_1 = P(X,X) and P_2 = P(Y,f(Y))
- P_1 and P_2 have no common substitution instance
- With σ={X/Y}
  - $P_1\sigma$ = P(X,X) and $P_2\sigma$ = P(X,f(X))
  - $P_1\sigma \neq P_2\sigma$
- With {f(Y)/X}
  - $P_1\sigma$ = P(f(Y),f(Y)) and $P_2\sigma$ = P(Y,f(Y))
  - $P_1\sigma \neq P_2\sigma$
- All other existing substitutions are similar
- No unifier can be found

Most General Unifier (MGU)

- MGU:
  - the most general substitution that unifies two expressions
  - the substitution that makes the least commitment
- Algorithm:
  - Unify(E1,E2)
    - if E1 and E2 are the same constant return {}
    - if E1 is a variable and E1 *does not occur in E2* return {E2/E1}
    - if E2 is a variable and E2 *does not occur in E1* return {E1/E2}
    - otherwise, we must have two functions or predicates
      - if the head matches and same arity
      - for each pair of terms t1, t2, call Unify(t1,t2) which returns a substitution S
      - apply S to the remaining terms and repeat the process for the remaining terms
      - return the total substitution (the composition) computed

Algorithm - Notation

- in the next algorithm, we use a list notation
- ex:
  - p(a,b) --> [p a b]
  - p(f(a), g(X,Y)) --> [p [f a] [g X Y]]
  - parents(X, father(X), mother(bill)) -->
    [parents X [father X] [mother bill]]

More Formally

```
function unify(E1, E2)
  if E1, E2 both constants or both empty then
    if E1 = E2 then return {} else return FAIL
  if E1 or E2 variable then
    if E1 variable
      if E1 occurs in E2 then return FAIL
      else return {E2/E1}
    else if E2 variable
      if E2 occurs in E1 then return FAIL
      else return {E1/E2}
  if E1 or E2 empty then return FAIL
  S1 := unify(first element of E1, first element of E2)
  if S1 = FAIL then return FAIL
  F1 := apply S1 to (rest of E1)
  F2 := apply S1 to (rest of E2)
  S2 := unify(F1, F2)
  if S2 = FAIL then return FAIL else return S1 ◦ S2
```

An Example

```
unify(  [parents X [father X] [mother bill]],
        [parents bill [father bill] Y] )

unify(parents, parents) --> {}
apply {} to rest of expression

unify(X [father X] [mother bill], bill [father bill] Y )
   unify(X, bill) --> {bill/X}
   apply {bill/X} --> [father bill] [mother bill], [father bill] Y

unify([father bill] [mother bill], [father bill] Y)
   unify(father, father) --> {}
   unify(bill, bill) --> {}
   unify([], []) --> {}
   unify([mother bill], Y) --> {[mother bill]/Y}
   apply {[mother bill]/Y} to rest of expression
   unify([], []) --> {}

return {bill/X}∘{[mother bill]/Y} = { bill/X, [mother bill]/Y }
```

Today

- Propositional Logics
- Predicate Logics
- Automated reasoning

  - Deduction
    - Forward Chaining
    - Backward Chaining
  - Resolution-Refutation

Automated Reasoning

- Automated reasoning (or automated theorem proving) is important for many tasks in AI:
  - reasoning in natural languages
  - automated proving of mathematical theorems
  - expert system shells
  - formal software verification
  - semantic web
  - ...

The Task

- We have:
  - A formal representation system (e.g., propositional logic, predicate logic) with a set of basic axioms
  - Knowledge base: facts and rules, expressed within that system
- We want:
  - to be able to know if a certain fact logically follows from the knowledge base
- We need:
  - automated proof procedure

Logical Inference

- Inference: process of deriving new facts from a set of premises

- Types of logical inference:
  1. Deduction
  2. Induction
  3. Abduction

Deduction

- aka Natural Deduction
- Conclusion follows necessary from the premises.
- We conclude from the general case to a specific example of the general case

- Ex:

  All men are mortal.
  Marcus is a men.

  Marcus is mortal.

- Sound inference

Induction

- Conclusion about all members of a class from the examination of only a few member of the class.
- We construct a general explanation based on a specific case.
- Ex:

  All CS students in COMP 472 are smart.

  All CS students on vacation are smart.

  All CS students are smart.

- Not sound
- But, can be seen as hypothesis construction or generalisation

Abduction

- Conclusion is one hypothetical (most probable) explanation for the premises
- From $A \Rightarrow B$ and the observation that A is true, we conclude that A is true
- Ex:

  Drunk people do not walk straight.
  John does not walk straight.

  John is drunk.

- Not sound… but may be most likely explanation for B
- Used in medicine…
  - in reality… disease $\Rightarrow$ symptoms
  - patient complains about some symptoms… doctor concludes a disease

Today

- Propositional Logics
- Predicate Logics
- Automated reasoning
  - Deduction
    - Forward Chaining
    - Backward Chaining
  - Resolution-Refutation

Rules of Deduction in Propositional Logics

1. Modus Ponens

$$\frac{P \Rightarrow Q \quad P}{Q}$$

2. Modus Tollens

$$\frac{\neg Q \quad P \Rightarrow Q}{\neg P}$$

3. And-Elimination

$$\frac{P \wedge Q}{P}$$

4. And-Introduction

$$\frac{P \quad Q}{P \wedge Q}$$

5. Or-Introduction

$$\frac{P}{P \vee Q}$$

6. Double Negation elimination

$$\frac{\neg\neg P}{P}$$

7. Unit Resolution

$$\frac{P \vee Q \quad \neg Q}{P}$$

8. Resolution

$$\frac{P \vee Q \quad \neg Q \vee R}{P \vee R}$$

$$\frac{\neg P \Rightarrow Q \quad Q \Rightarrow R}{\neg P \Rightarrow R}$$

Rules of Deduction in Predicate Logics

- Deduction rules of propositional logics
- Universal Elimination:
  - g= ground term (no var)

$$\frac{\forall X \;\; P(X)}{P(g)} \qquad \frac{\forall X \;\; likes(X, iceCream)}{likes(mary, iceCream)}$$

- Existential Elimination
  (i.e. naming an existential var.)
  - k=a new constant

$$\frac{\exists X \;\; P(X)}{P(k)} \qquad \frac{\exists X \; kill(X, Y)}{kill(murderer, Y)}$$

- Existential Introduction
  - X=variable not appearing in P

$$\frac{P(g)}{\exists X \;\; P(X)} \qquad \frac{likes(jerry, iceCream)}{\exists X \; likes(X, iceCream)}$$

Proof Procedure

- A proof procedure is a combination of
  - an inference rule, and
  - an algorithm for applying the rule to a set of expressions
- Assume:
  - we have the facts $F_1, F_2, \ldots, F_n$
  - we want to show that G is a logical consequence.
  - **Direct proof:**
    - show that $(F_1 \wedge F_2 \wedge \ldots \wedge F_n) \rightarrow G$ is a tautology
  - **Refutation proof:**
    - show that $(F_1 \wedge F_2 \wedge \ldots \wedge F_n \wedge \neg G)$ leads to an inconsistency

Example of Direct Proof (Deduction)

- Premises (facts in the KB):
  1. André is in top form.
  2. André enters the tennis tournament
  3. If André is in top form and he plays tennis, then the spectators enjoy the game.
  4. If André enters the tennis tournament, then he plays tennis.
- Goal (to prove)
  - The spectators enjoy the game.

Example of Direct Proof (Deduction)

1. Encode NL into propositional logic

   A: André is in top form.
   B: André enters the tennis tournament.
   C: André plays tennis.
   D: The spectators enjoy the game.

2. Use inference rules to deduce new facts

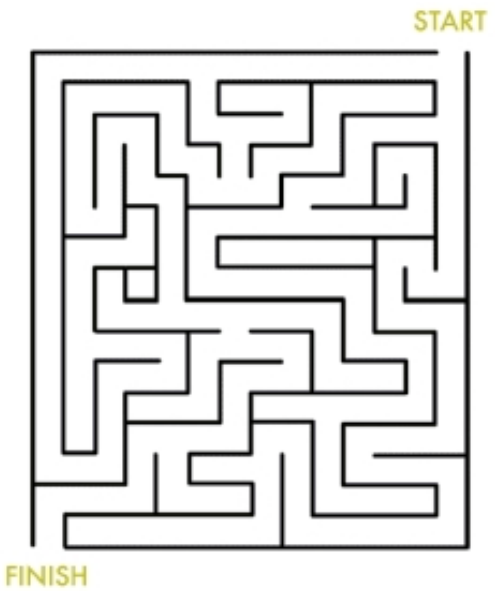
1.	**A**	from KB
2.	**B**	from KB
3.	**$(A \wedge C) \Rightarrow D$**	from KB
4.	**$B \Rightarrow C$**	from KB
5.	**C**	modus ponens on [2] & [4]
6.	**$(A \wedge C)$**	and-introduction on [1] & [5]
7.	**D**	modus ponens on [3] & [6]

Today

- Propositional Logics
- Predicate Logics
- Automated reasoning
  - Deduction
    - Forward Chaining
    - Backward Chaining
  - Resolution-Refutation

Algorithms for Deduction

- Forward chaining:
  - --> from the data to the goal
  - --> from a new data to its consequence



- Backward chaining:
  - --> try to prove what could prove the goal
  - --> from the goal to the premises

Forward Chaining

- When a new fact P is added to the KB
- For each rule such that P unifies with a premise
  - If the other premises are known
  - Then add the conclusion to the KB and continue chaining

- Forward chaining is data-driven
- Infer all new knowledge each time a sentence is added to the KB
- Susceptible to creating many irrelevant facts

Eg. Deduction + Forward Chaining (1)

Assume the KB:

1. man(marcus)
2. fromPompae(marcus)
3. $\forall$X fromPompae(X) $\Rightarrow$ roman(X)
4. leader(caesar)
5. $\forall$X roman(X) $\Rightarrow$ (loyalTo(X,caesar) $\vee$ hated(X,caesar))
6. $\forall$X $\exists$y loyalTo(x,y)
7. $\forall$X $\forall$Y man(X) $\wedge$ leader(Y) $\wedge$ tryToAssasinate(X,Y) $\Rightarrow$ $\neg$loyalTo(X,Y)
8. tryToAssasinate(marcus,caesar)

Eg. Deduction + Forward Chaining (1)

Goal to prove : **hate(marcus,caesar)**

Modus ponens [2]+[3] with {**marcus/X**}
[9] **roman(marcus)**

Modus ponens [5]+[9] with {**marcus/X**}
[10] **loyalTo(marcus,caesar) ∨ hated(marcus,caesar)**

Modus ponens [1]+[4]+[8]+[7] with {marcus/X,caesar/Y}
[11] **¬loyalTo(x,y)**

Resolution on [10]+[11]
hated(marcus,caesar)

Today

- Propositional Logics
- Predicate Logics
- Automated reasoning
  - Deduction
    - Forward Chaining
    - Backward Chaining

  - Resolution-Refutation

Backward Chaining

- When a query Q is asked
  - If a matching fact Q' is known, return the unifier
  - For each rule whole consequent Q' unifies Q, attempt to prove each premise of the rule by backward chaining.

- Backward chaining is the basis for logic programming (Prolog)
- Natural for proving goals

- More directed and efficient - Tries to infer only useful information (for the current goal)

Eg. Deduction + Backward Chaining (1)

Assume the KB:

1. man(marcus)
2. fromPompae(marcus)
3. $\forall$X fromPompae(X) $\Rightarrow$ roman(X)
4. leader(caesar)
5. $\forall$X roman(X) $\Rightarrow$ (loyalTo(X,caesar) $\vee$ hated(X,caesar))
6. $\forall$X $\exists$Y loyalTo(X,Y)
7. $\forall$X $\forall$Y man(X) $\wedge$ leader(Y) $\wedge$ tryToAssasinate(X,Y) $\Rightarrow$ $\neg$loyalTo(X,Y)
8. tryToAssasinate(marcus,caesar)

Eg. Deduction + Backward Chaining (1)

Goal to prove : $\neg$`loyalTo(marcus,caesar)`

[7] with {`marcus/X, caesar/Y`}
Sub-goal: `man(marcus)` $\wedge$
`leader(ceasar)` $\wedge$
`tryToAssasinate(marcus,ceasar)`

[4]
Sub-goal: `man(marcus)` $\wedge$
`tryToAssasinate(marcus,ceasar)`

[8]
Sub-goal: `man(marcus)`

[1]
Sub-goal: $\varnothing$

Today

- Propositional Logics
- Predicate Logics
- Automated reasoning
  - Deduction
    - Forward Chaining
    - Backward Chaining
  - Resolution Refutation

Remember this Rule of Deduction

1. Modus Ponens

$$\frac{P \Rightarrow Q \quad P}{Q}$$

2. Modus Tollens

$$\frac{\neg Q \quad P \Rightarrow Q}{\neg P}$$

3. And-Elimination

$$\frac{P \wedge Q}{P}$$

4. And-Introduction

$$\frac{P \quad Q}{P \wedge Q}$$

5. Or-Introduction

$$\frac{P}{P \vee Q}$$

6. Double Negation elimination

$$\frac{\neg\neg P}{P}$$

7. Unit Resolution

$$\frac{P \vee Q \quad \neg Q}{P}$$

8. Resolution

$$\frac{P \vee Q \quad \neg Q \vee R}{P \vee R}$$

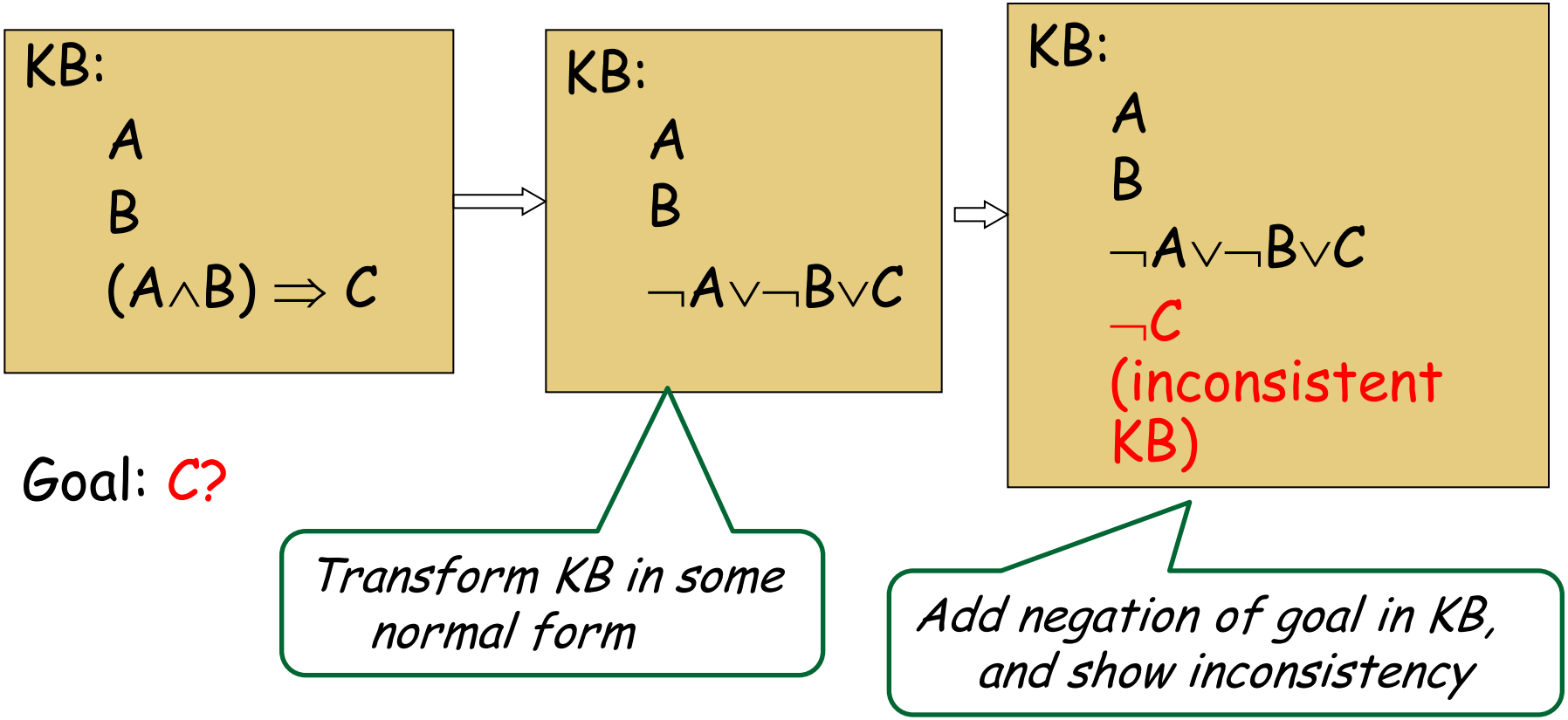
$$\frac{\neg P \Rightarrow Q \quad Q \Rightarrow R}{\neg P \Rightarrow R}$$

Proof by Resolution-Refutation

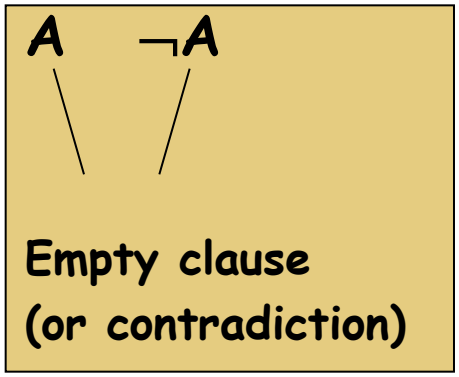
- With natural deduction it is not clear *how* the process can be automated
  - each step requires some initiative to determine the right next step

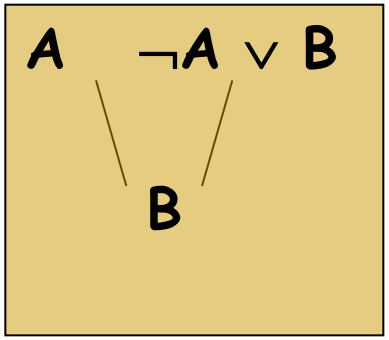
- Resolution Refutation can be automated because it involves a fixed set of steps.

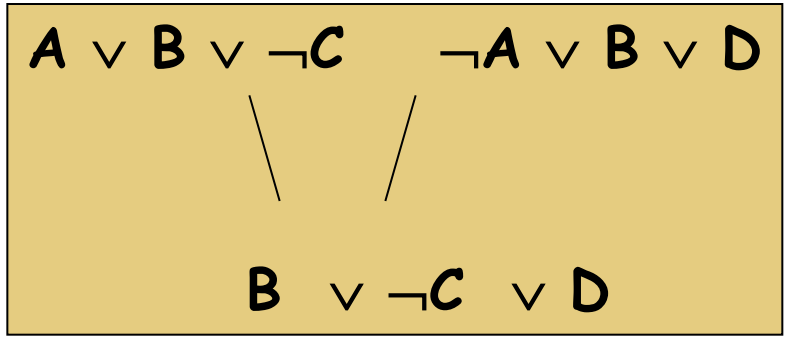
Resolution Refutation: Intuition



The Basis of Resolution Refutation







More Formally

$$\text{resolution}: \frac{P \vee Q \qquad \neg Q \vee R}{P \vee R}$$

- To use this rule, we must have facts as a conjunction of disjunctions…
- Resolution works on a standardized logical form
  - conjunctive normal form (CNF) or
  - implicative normal form (INF)

- The mechanics of resolution:
  1. Transform clauses into CNF (or INF)
  2. Add the negation of what we wish to prove in the KB
  3. *Resolve* pairs of clauses by producing new clauses until
     - you have produced an empty clause (contraction… goal is proven)
     - or, no more pairs of clauses can be resolved (failure to prove)

Resolution Refutation in Predicate Logics

- Similar to resolution-refutation in propositional logics, but:
  - Must deal with quantifiers
  - Must deal with substitutions

Conversion to CNF

- CNF - Conjunctive Normal Form = conjunction of disjunctions
- Any predicate logic formula can be transformed into CNF

Ex: $\forall X\ (p(X) \Rightarrow (\forall Y(p(Y) \Rightarrow p(f(X,Y))) \wedge \neg\forall Y(q(X,Y) \Rightarrow p(Y))))$

-->

$\neg p(X) \vee \neg p(Y) \vee p(f(X,Y)$

$\neg p(X) \vee q(X,g(X))$

$\neg p(X) \vee \neg p(g(X))$

Conversion to CNF (1)

- Running example:
  - $\forall X\ (\forall Y\ p(a,Y) \wedge s(Y,X)) \Rightarrow \neg(\forall Y\ q(X,Y) \wedge s(Y,X) \Rightarrow r(X,Y))$

- **Step 1**: remove " $\Rightarrow$" and "$\Leftrightarrow$"
  - replace $A \Rightarrow B$ by $\neg A \vee B$
  - replace $A \Leftrightarrow B$ by $(\neg A \vee B) \wedge (A \vee \neg B)$

- Example after step 1
  - $\forall X\ (\forall Y\ p(a,Y) \wedge s(Y,X)) \Rightarrow \neg(\forall Y\ q(X,Y) \wedge s(Y,X) \Rightarrow r(X,Y))$
  - $\forall X\ \neg(\forall Y\ p(a,Y) \wedge s(Y,X)) \vee \neg(\forall Y\ \neg(q(X,Y) \wedge s(Y,X)) \vee r(X,Y))$

Conversion to CNF (2)

- **Step 2**: reduce scope of negation
  - replace $\neg\neg A$ by A
  - replace $\neg(A \vee B)$ by $\neg A \wedge \neg B$
  - replace $\neg(A \wedge B)$ by $\neg A \vee \neg B$
  - replace $\neg(\forall X\ A)$ by $(\exists X\ \neg A)$
  - replace $\neg(\exists X\ A)$ by $(\forall X\ \neg A)$

- Example after step 2
  - $\forall X\ \underline{\neg(\forall Y\ p(a,Y) \wedge s(Y,X))} \vee \underline{\neg(\forall Y\ \neg(q(X,Y) \wedge s(Y,X)) \vee r(X,Y))}$
  - $\forall X\ (\exists Y\ \neg p(a,Y) \vee \neg s(Y,X)) \vee (\exists Y\ q(X,Y) \wedge s(Y,X) \wedge \neg r(X,Y))$

Conversion to CNF (3)

- **Step 3**: rename variables such that no 2 quantifiers bind the same variable

- Example after step 3
  - $\forall X\ (\exists Y\ \neg p(a,Y) \vee \neg s(Y,X)) \vee (\exists Y\ q(X,Y) \wedge s(Y,X) \wedge \neg r(X,Y))$
  - $\forall X\ (\exists Y\ \neg p(a,Y) \vee \neg s(Y,X)) \vee (\exists Z\ q(X,Z) \wedge s(Z,X) \wedge \neg r(X,Z))$

Conversion to CNF (4)

- **Step 4**: (prenex normal form) move all quantifiers to the left without changing their order

- Example after step 4
  - $\forall X\ (\exists Y\ \neg p(a,Y) \vee \neg s(Y,X)) \vee (\exists Z\ q(X,Z) \wedge s(Z,X) \wedge \neg r(X,Z))$
  - $\forall X\ \exists Y\ \exists Z\ (\neg p(a,Y) \vee \neg s(Y,X)) \vee ((q(X,Z) \wedge s(Z,X) \wedge \neg r(X,Z)))$

Conversion to CNF (5a)

- **Step 5**: Skolemization: replace each existentially quantified variable by a skolem constant or skolem function and remove the quantifier ∃.

A Note on Skolemization

- Named after the Polish logician Skolem
- Goal: give a name to existentially quantified variables
- Replace existentially quantified variables with:
  - *case a:* a new constant

  or
  - *case b:* a new function where its arguments include all universally quantified variables scoped outside of it

Conversion to CNF (5a)

Assume that X is existentially quantified:

- ***case a:*** replace X by a unique, fresh skolem constant, if X is not universally quantified
- ex:
  - $\exists X\ p(X) \rightarrow p(sk)$
  - $\exists X\ \forall Y\ p(X,Y,X) \rightarrow \forall Y\ p(sk, Y, sk)$
  - $\exists X\ \exists Y\ \forall Z\ p(X,Y,Z) \rightarrow \forall Z\ p(sk1, sk2, Z)$

Conversion to CNF (5b)

- ***case b:*** if an existentially quantified variable is within the scope of one (or several) universally quantified variables, then replace it with a skolem function of this (these) variable(s)
- ex:
  - $\forall X\ \exists Y\ p(X,Y) \rightarrow \forall X\ p(X,\ sk1(X)\ )$
  - $\forall X\ \forall Y\ \exists Z\ p(X,Y,Z) \rightarrow \forall X \forall Y\ p(X,\ Y,\ sk2(X,Y)\ )$
  - $\forall X\ \exists Y\ \exists Z\ p(X,Y) \wedge q(X,Z) \rightarrow$

    $\forall X\ p(X,\ sk3(X)\ ) \wedge q(X,\ sk4(X)\ )$

Conversion to CNF (5)

- Example after step 5 (skolemization)
  - $\forall X\ \exists Y\ \exists Z\ (\neg p(a,Y) \vee \neg s(Y,X)) \vee$
    $(q(X,Z) \wedge s(Z,X) \wedge \neg r(X,Z))$
  - $\forall X\ \neg p(a, sk1(X)) \vee \neg s(sk1(X),X) \vee$
    $(q(X, sk2(X)) \wedge s(sk2(X),X) \wedge \neg r(X,sk2(X)))$

Conversion to CNF (6)

- **Step 6**: Drop all universal quantifiers.

  Now all variables are universally quantified and the universal quantifiers can be omitted

- Example after step 6
  - $\forall X \neg p(a, sk1(X)) \vee \neg s(sk1(X), X) \vee$

    $(q(X, sk2(X)) \wedge s(sk2(X), X) \wedge \neg r(X, sk2(X)))$
  - $\neg p(a, sk1(X)) \vee \neg s(sk1(X), X) \vee$

    $(q(X, sk2(X)) \wedge s(sk2(X), X) \wedge \neg r(X, sk2(X)))$

Conversion to CNF (7)

- **Step 7**: transform into conjunctive normal form, i.e., a conjunction of disjunctions
  - apply the distributive law: $P \vee (Q \wedge R) \equiv (P \vee Q) \wedge (P \vee R)$
  - each conjunction is called a clause

- Example after step 7
  - $\neg p(a, sk1(X)) \vee \neg s(sk1(X),X) \vee$

    $(q(X, sk2(X)) \wedge s(sk2(X),X) \wedge \neg r(X, sk2(X)))$
  - $(\neg p(a, sk1(X)) \vee \neg s(sk1(X),X) \vee q(X, sk2(X))) \wedge$

    $(\neg p(a, sk1(X)) \vee \neg s(sk1(X),X) \vee s(sk2(X),X)) \wedge$

    $(\neg p(a, sk1(X)) \vee \neg s(sk1(X),X) \vee \neg r(X,sk2(X)))$

Conversion to CNF (8)

- **Step 8**: Rewrite as a set of clauses, where $\wedge$ is considered to be the default between clauses
  - ex: $(P \vee Q) \wedge (R \vee S) \wedge (R \vee S \vee T)$ is written as
    - $P \vee Q$
    - $R \vee S$
    - $R \vee S \vee T$
- Example after step 8
  - $(\neg p(a, sk1(X)) \vee \neg s(sk1(X),X) \vee q(X, sk2(X))) \wedge$
    $(\neg p(a, sk1(X)) \vee \neg s(sk1(X),X) \vee s(sk2(X),X)) \wedge$
    $(\neg p(a, sk1(X)) \vee \neg s(sk1(X),X) \vee \neg r(X, sk2(X)))$
  - $\neg p(a, sk1(X)) \vee \neg s(sk1(X), X) \vee q(X, sk2(X))$
  - $\neg p(a, sk1(X)) \vee \neg s(sk1(X), X) \vee s(sk2((X), X)$
  - $\neg p(a, sk1(X)) \vee \neg s(sk1(X), X) \vee \neg r(X, sk2((X))$

Conversion to CNF (9)

- **Step 9**: Standardize variables apart.

  all variables occuring in all clauses are renamed such that no 2 clauses will share the same variable

- Example after step 9
  - $\neg p(a, sk1(X)) \vee \neg s(sk1(X), X) \vee q(X, sk2(X))$
  - $\neg p(a, sk1(X)) \vee \neg s(sk1(X), X) \vee s(sk2(X), X)$
  - $\neg p(a, sk1(X)) \vee \neg s(sk1(X), X) \vee \neg r(X, sk2(X))$
  - $\neg p(a, sk1(X)) \vee \neg s(sk1(X), X) \vee q(X, sk2(X))$
  - $\neg p(a, sk1(Y)) \vee \neg s(sk1(Y), Y) \vee s(sk2(Y), Y)$
  - $\neg p(a, sk1(Z)) \vee \neg s(sk1(Z), Z) \vee \neg r(X, sk2(Z))$

Summary of Transformations

- **Step 1**: Remove $\Rightarrow$ and $\Leftrightarrow$
- **Step 2**: Reduce scope of negations (negation normal form)
- **Step 3**: Rename variables in such a way that no 2 quantifiers bind the same variable
- **Step 4**: Move quantifiers to the left (prenex normal form)
- **Step 5**: Replace each existentially quantified variable by a skolem constant or skolem function and remove the existential quantifier ($\exists$)
- **Step 6**: Drop universal quantifiers
- **Step 7**: Transformation into conjunctive normal form
- **Step 8**: Rewrite as a set of clauses, where $\wedge$ is considered to be the default between clauses
- **Step 9**: Rename variables so that no 2 clauses share the same variable

Example of Resolution-Refutation

Jack owns a dog.

Every dog owner is an animal lover.

No animal lover kills an animal.

Either Jack or Curiosity killed the cat, who, by the way, is named Tuna.

Did Curiosity kill the cat?

1. $\exists X$ `dog(X)` $\wedge$ `owns(jack,X)`
2. $\forall X$ ($\exists Y$ `dog(Y)` $\wedge$ `owns(X,Y)` $\Rightarrow$ `animalLover(X)`)
3. $\forall X$ `animalLover(X)` $\Rightarrow$ ($\forall Y$ `animal(Y)` $\Rightarrow$ $\neg$`kills(X,Y)`)
4. `kills(jack,tuna)` $\vee$ `kills(curiosity,tuna)`
5. `cat(tuna)`
6. $\forall X$ `cat(X)` $\Rightarrow$ `animal(X)` from world knowledge

Prove that curiosity killed the cat: `kills(curiosity,tuna)`

Conversion to CNF

1a. dog(sk)

1b. owns(jack, sk)

2. ¬dog(f(X)) ∨ ¬owns(X, f(X)) ∨ animalLover(X)

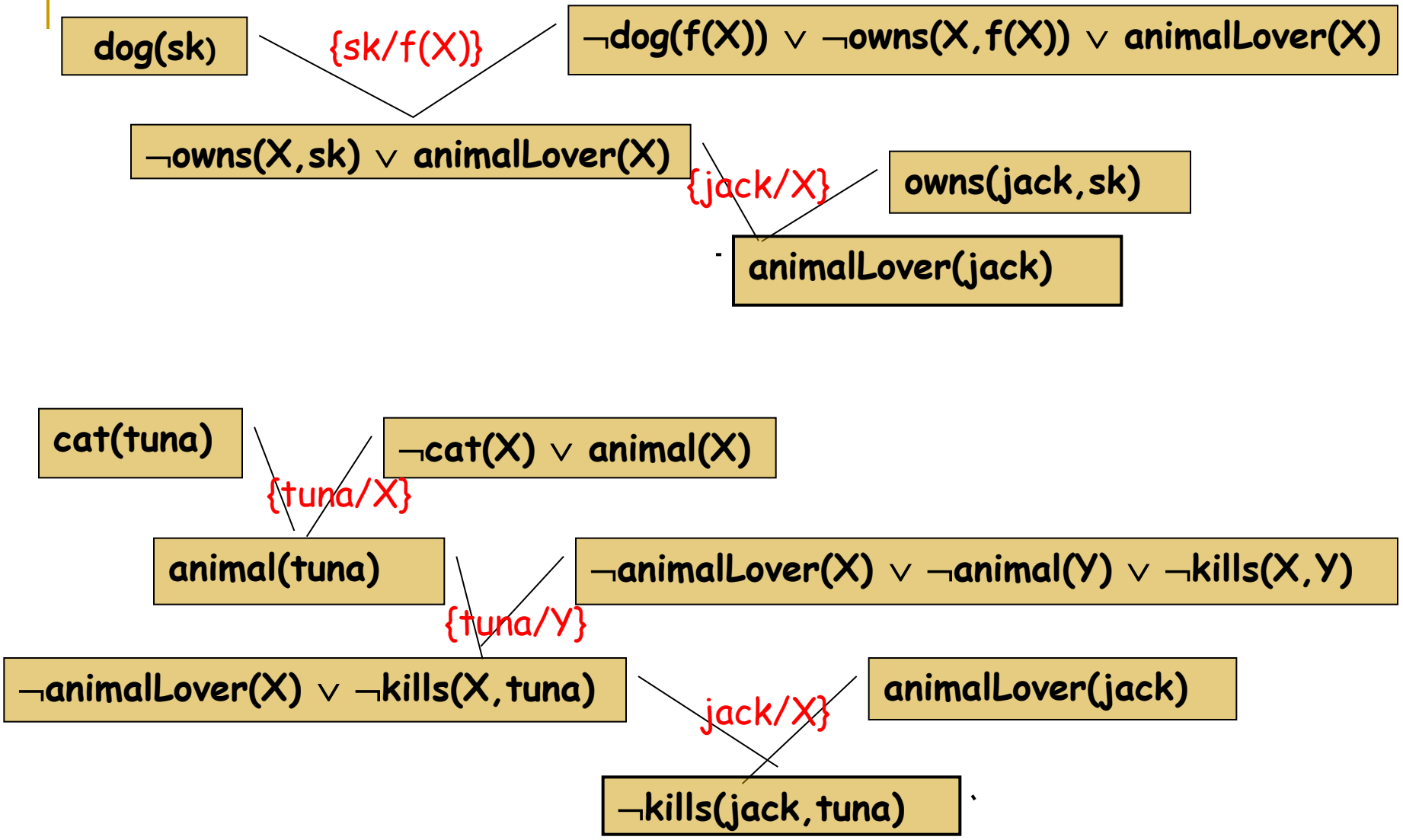
3. ¬animalLover(X) ∨ ¬animal(Y) ∨ ¬kills(X,Y)

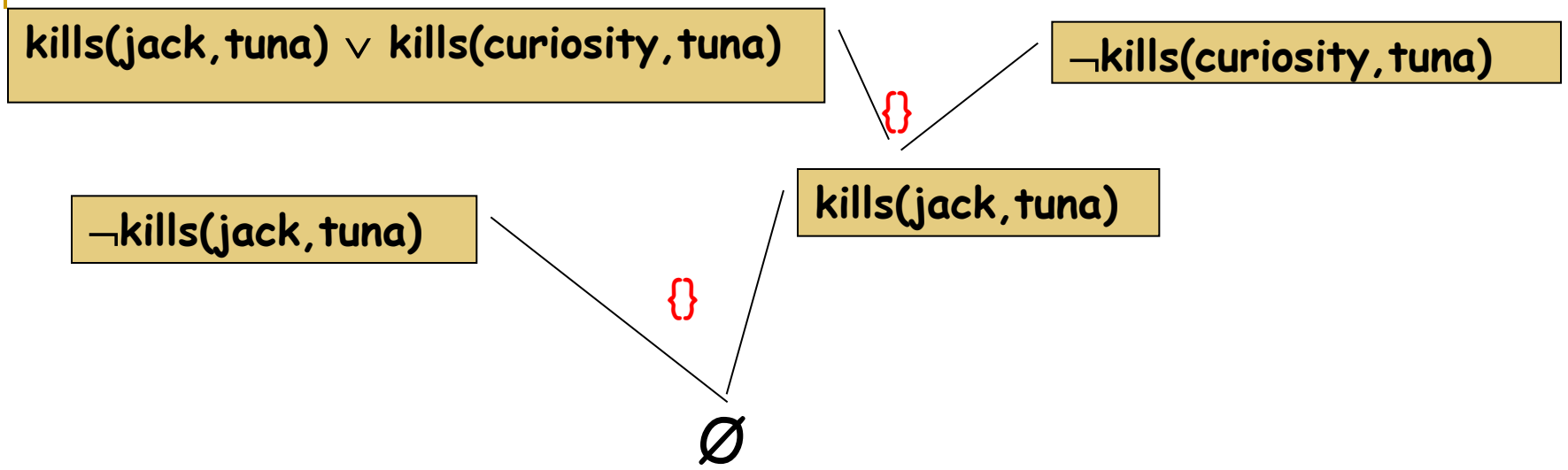
4. kills(jack,tuna) ∨ kills(curiosity,tuna)

5. cat(tuna)

6. ¬cat(X) ∨ animal(X)

7. Add: ¬kills(curiosity,tuna)


dog(sk)
{sk/f(X)}
¬dog(f(X)) ∨ ¬owns(X,f(X)) ∨ animalLover(X)
¬owns(X,sk) ∨ animalLover(X)
{jack/X}
owns(jack,sk)
animalLover(jack)
cat(tuna)
¬cat(X) ∨ animal(X)
{tuna/X}
animal(tuna)
¬animalLover(X) ∨ ¬animal(Y) ∨ ¬kills(X,Y)
{tuna/Y}
¬animalLover(X) ∨ ¬kills(X,tuna)
jack/X}
animalLover(jack)
¬kills(jack,tuna)


kills(jack,tuna) ∨ kills(curiosity,tuna)
¬kills(curiosity,tuna)
{}
kills(jack,tuna)
¬kills(jack,tuna)
{}
∅

Resolution: Complexity Issues

- Two clauses can have several resolvents
  - C_1: `bunny(bugs) ∨ bunny(roger)`
  - C_2: `¬bunny(X) ∨ likes(X,carrots)`

- Two resolvents of C_1 and C_2
  - C_3: `bunny(bugs) ∨ likes(roger,carrots)` `{roger/X}`
  - C_4: `bunny(roger) ∨ likes(bugs,carrots)` `{bugs/X}`

Resolution: Complexity Issues

- Resolution is can be seen as a search problem, so what is the branching factor **b**?
  - **N** clauses in knowledge base
  - in the first resolution step, we have N^2 ways of combining them and
  - generate new clauses and add them to the knowledge base
  - overall, exponential complexity
- Thus, we have to apply heuristic search

Resolution Heuristics (1)

- Breadth-first search:
  - uninformed search
  - complete
- Subsumption:
  - Keep the initial KB as small as possible by removing clauses that are more specific than others (subsumed by already existing clauses)
  - ex: if KB contains p(X), we can remove p(a)
  - complete
- Set of support:
  - Use a set of clauses from the KB and always resolve clauses from this set with clauses not in the set and add the resolvent in the set.
  - complete, if the right subset is chosen

Resolution Heuristics (2)

- Unit preference strategy:
  - idea: we want shorter clauses; the empty result is the shortest with size 0
  - prefer resolutions that produce shorter sentences
  - an easy way to achieve this is to use unit clauses having only one literal
  - the result of a resolution with a unit is always shorter than the parent
  - incomplete
  - useful if combined with other heuristics

Resolution Heuristics (3)

- Linear input form strategy:
  - idea: reduce the search space by only allowing resolutions with original sentences
  - start with negated goal and resolve it with one of the original sentences
  - result is again resolved with one of the original sentences, never with another intermediary result
  - incomplete

Binary Resolution vs Hyperresolution

- what we have seen so far
- exactly 2 parent clauses are clashed

- Another method is to combine several parent clauses in a single step
- This is called hyperresolution
- ex:
  - ¬married(X,Y) ∨ ¬mother(X,Z) ∨ father(Y,Z)
  - married(kate,george) ∨ likes(george,kate)
  - mother(kate,sara)

  - father(george,sara) ∨ likes(george,kate)

Logic Programming with Prolog

- Logic Programming is probably the most important implementation of resolution
  - only simplified clauses (Horn clauses), of the form
    $a \leftarrow b_1 \wedge b_2 \wedge \ldots \wedge b_n = a \vee \neg b_1 \vee \neg b_2 \vee \ldots \vee \neg b_n$
    are allowed
  - built-in resolution inferencing with backward chaining, depth-first search and several heuristics
  - left-to-right search through conjuncts, first-to-last search in knowledge base, no occur-check in unification
  - negation as failure: $\neg p$ is considered proved if no proof for p is found
    - closed-world assumption

What about these statements?

- How do you represent:
  - Blond-haired people often have blue eyes
  - Mary is slightly ill
  - It might rain tonight
  - Mary believes that John loves her
  - I know that you know that I know that you know I will be leaving town tomorrow
  - …

Other types of Logics

- Fuzzy logic:
  - degree of belonging to the set of *hot days*, *ill people*, ...
- Higher-order logic
  - quantifiers can bind predicates and predicates can take other predicates as arguments
- Temporal logics
  - allows reasoning about time
- Modal logics
  - allows the expression of possibility, necessity, belief, knowledge, temporal progression …
- Description logics
  - used in the Semantic Web

Today

- Propositional Logics
- Predicate Logics
- Automated reasoning
  - Deduction
    - Forward Chaining
    - Backward Chaining
  - Resolution Refutation