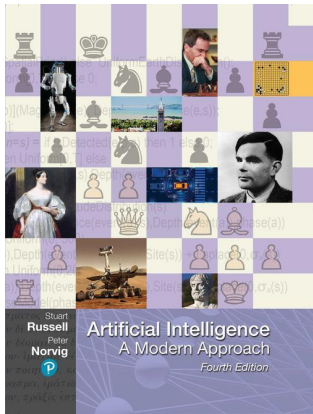


# Artificial Intelligence: A Modern Approach

Fourth Edition



## Chapter 9

Inference in first-order logic

### Outline

- Reducing first-order inference to propositional inference
- Unification
- Generalized Modus Ponens
- Forward and backward chaining
- Logic programming
- Resolution

### Universal instantiation (UI)

Every instantiation of a universally quantified sentence is entailed by it:

$$\forall v \quad a$$

$$\text{Subst}(\{v/g\}, a)$$

for any variable  $v$  and ground term  $g$

E.g.,  $\forall x \quad \text{King}(x) \wedge \text{Greedy}(x) \Rightarrow \text{Evil}(x)$  yields

$$\text{King}(J \text{ ohn}) \wedge \text{Greedy}(J \text{ ohn}) \Rightarrow \text{Evil}(J \text{ ohn})$$

$$\text{King}(\text{Richard}) \wedge \text{Greedy}(\text{Richard}) \Rightarrow$$

$$\text{Evil}(\text{Richard})$$

$$\text{King}(\text{Father}(J \text{ ohn})) \wedge \text{Greedy}(\text{Father}(J \text{ ohn})) \Rightarrow \text{Evil}(\text{Father}(J \text{ ohn}))$$

.

### Existential instantiation (EI)

For any sentence  $a$ , variable  $v$ , and constant symbol  $k$  that does not appear elsewhere in the knowledge base:

$$\exists v \quad a$$

$$\text{Subst}(\{v/k\}, a)$$

E.g.,  $\exists x \quad \text{Crown}(x) \wedge \text{OnHead}(x, J \text{ ohn})$  yields

$$\text{Crown}(C_1) \wedge \text{OnHead}(C_1, J \text{ ohn})$$

provided  $C_1$  is a new constant symbol, called a **Skolem**

**constant** Another example: from  $\exists x \quad d(x^y)/dy = x^y$  we obtain

$$d(e^y)/dy = e^y$$

provided  $e$  is a new constant symbol

### Existential instantiation contd.

UI can be applied several times to **add** new sentences; the new KB is logically equivalent to the old

EI can be applied once to **replace** the existential sentence; the new KB is **not** equivalent to the old, but is satisfiable iff the old KB was satisfiable

### Reduction to propositional inference

Suppose the KB contains just the following:

$$\forall x \quad \text{King}(x) \wedge \text{Greedy}(x) \Rightarrow$$

$$\text{Evil}(x) \quad \text{King}(J \text{ ohn})$$

$$\text{Greedy}(J \text{ ohn})$$

$$\text{Brother}(\text{Richard}, J$$

$$\text{ohn})$$

Instantiating the universal sentence in **all possible** ways, we have

$$\text{King}(J \text{ ohn}) \wedge \text{Greedy}(J \text{ ohn}) \Rightarrow \text{Evil}(J \text{ ohn})$$

$$\text{King}(\text{Richard}) \wedge \text{Greedy}(\text{Richard}) \Rightarrow$$

$$\text{Evil}(\text{Richard}) \quad \text{King}(J \text{ ohn})$$

$$\text{Greedy}(J \text{ ohn})$$

$$\text{Brother}(\text{Richard}, J$$

$$\text{ohn})$$

The new KB is **propositionalized**: proposition symbols are

## Reduction contd.

Claim: a ground sentence\* is entailed by new KB iff entailed by original KB

Claim: every FOL KB can be propositionalized so as to preserve

entailment Idea: propositionalize KB and query, apply resolution, return

result Problem: with function symbols, there are infinitely many ground

terms,

e.g.,  $Father(Father(Father(J\ ohn)))$


Theorem: Herbrand (1930). If a sentence  $a$  is entailed by an FOL KB, it is entailed by a **finite** subset of the propositional KB

Idea: For  $n = 0$  to  $\infty$  do

create a propositional KB by instantiating with depth- $n$

terms see if  $a$  is entailed by this KB

Problem: works if  $a$  is entailed, loops if  $a$  is not entailed

 Turing (1936), Church (1936), entailment in FOL is **semidecidable**

## Problems with propositionalization

Propositionalization seems to generate lots of irrelevant sentences. E.g., from

$\forall x King(x) \wedge Greedy(x) \Rightarrow$   
 $Evil(x) \quad King(J\ ohn)$   
 $\forall y \quad Greedy(y)$   
 $Brother(Richard, J$   
 $ohn)$

it seems obvious that  $Evil(J\ ohn)$ , but propositionalization produces lots of facts such as  $Greedy(Richard)$  that are irrelevant  
 With  $p$   $k$ -ary predicates and  $n$  constants, there are  $p \cdot n^k$

instantiations With function symbols, it gets much much worse!

 Pearson

© 2021 Pearson Education Ltd.

Chapter 9 8

## Unification

We can get the inference immediately if we can find a substitution  $\theta$  such that  $King(x)$  and  $Greedy(x)$  match  $King(J\ ohn)$  and  $Greedy(y)$

$\theta = \{x/J\ ohn, y/J\ ohn\}$  works

$Unify(a, \beta) = \theta$  if  $a\theta = \beta\theta$

$p$	$q$	$\theta$
$Knows(J\ ohn, x)$	$Knows(J\ ohn, Jane)$	
$Knows(J\ ohn, x)$	$Knows(y, OJ)$	
$Knows(J\ ohn, x)$	$Knows(y, M\ other(y))$	
$Knows(x, OJ)$		

## Unification

We can get the inference immediately if we can find a substitution  $\theta$  such that  $King(x)$  and  $Greedy(x)$  match  $King(J\ ohn)$  and  $Greedy(y)$

$\theta = \{x/J\ ohn, y/J\ ohn\}$  works

$Unify(a, \beta) = \theta$  if  $a\theta = \beta\theta$

$p$	$q$	$\theta$
$Knows(J\ ohn, x)$	$Knows(J\ ohn, Jane)$	$\{x/J\ ohn\}$
$Knows(J\ ohn, x)$	$Knows(y, OJ)$	
$Knows(J\ ohn, x)$	$Knows(y, M\ other(y))$	
$Knows(x, OJ)$		

 Pearson

© 2021 Pearson Education Ltd.

Chapter 9 9

 Pearson

© 2021 Pearson Education Ltd.

Chapter 9 10

## Unification

We can get the inference immediately if we can find a substitution  $\theta$  such that  $King(x)$  and  $Greedy(x)$  match  $King(J\ ohn)$  and  $Greedy(y)$

$\theta = \{x/J\ ohn, y/J\ ohn\}$  works

$Unify(a, \beta) = \theta$  if  $a\theta = \beta\theta$

$p$	$q$	$\theta$
$Knows(J\ ohn, x)$	$Knows(J\ ohn, Jane)$	$\{x/J\ ohn\}$
$Knows(J\ ohn, x)$	$Knows(y, OJ)$	$\{x/J\ ohn, y/J\ ohn\}$
$Knows(J\ ohn, x)$	$Knows(y, M\ other(y))$	
$Knows(x, OJ)$		

## Unification

We can get the inference immediately if we can find a substitution  $\theta$  such that  $King(x)$  and  $Greedy(x)$  match  $King(J\ ohn)$  and  $Greedy(y)$

$\theta = \{x/J\ ohn, y/J\ ohn\}$  works

$Unify(a, \beta) = \theta$  if  $a\theta = \beta\theta$

$p$	$q$	$\theta$
$Knows(J\ ohn, x)$	$Knows(J\ ohn, Jane)$	$\{x/J\ ohn\}$
$Knows(J\ ohn, x)$	$Knows(y, OJ)$	$\{x/J\ ohn, y/J\ ohn\}$
$Knows(J\ ohn, x)$	$Knows(y, M\ other(y))$	
$Knows(x, OJ)$		

 Pearson

© 2021 Pearson Education Ltd.

Chapter 9 11

 Pearson

© 2021 Pearson Education Ltd.

Chapter 9 12

## Unification

We can get the inference immediately if we can find a substitution  $\theta$  such that  $King(x)$  and  $Greedy(x)$  match  $King(John)$  and  $Greedy(y)$

$\theta = \{x/John, y/John\}$  works

Unify( $a, \beta$ ) =  $\theta$  if  $a\theta = \beta\theta$

$p$	$q$	$\theta$
$Knows(John, x)$	$Knows(John, Jane)$	$\{x/Jane\}$
$Knows(John, x)$	$Knows(y, Jane)$	$\{x/OJ, y/J\}$
$Knows(John, x)$	$Knows(y, Mother(y))$	$\{y/John, x/Mother(John)\}$
$Knows(John, x)$	$Knows(x, OJ)$	fail

Standardizing apart eliminates overlap of variables, e.g.,  $Knows(z_1, OJ)$

## Generalized Modus Ponens (GMP)

$$\frac{p_1^1, p_2^1, \dots, p_n^1}{p_n^1} \frac{(p_1 \wedge p_2 \wedge \dots \wedge p_n) \Rightarrow q}{\theta} \text{ where } p_i^1 \theta = p_i \text{ for all } i$$

$p_1^1$  is  $King(John)$   $p_1$  is  $King(x)$   
 $p_2^1$  is  $Knows(John, y/John)$   $p_2$  is  $Knows(x, y)$   
 $p_n^1$  is  $Greedy(y)$   $p_n$  is  $Greedy(x)$   
 $q\theta$  is  $Evil(John)$

GMP used with KB of definite clauses (exactly one positive literal)

All variables assumed universally quantified

## Soundness of GMP

Need to show that

$$p_1^1, \dots, p_n^1, (p_1 \wedge \dots \wedge p_n \Rightarrow q) \models q\theta$$

provided that  $p_i^1 \theta = p_i$  for all  $i$

Lemma: For any definite clause  $p$ , we have  $p \models p\theta$  by UI

- $(p_1 \wedge \dots \wedge p_n \Rightarrow q) \models (p_1 \wedge \dots \wedge p_n \Rightarrow q)\theta = (p_1\theta \wedge \dots \wedge p_n\theta \Rightarrow q\theta)$
- $p_1^1, \dots, p_n^1 \models p_1^1 \wedge \dots \wedge p_n^1 \models p_1\theta \wedge \dots \wedge p_n\theta$
- From 1 and 2,  $q\theta$  follows by ordinary Modus Ponens

## Example knowledge base

The law says that it is a crime for an American to sell weapons to hostile nations. The country Nono, an enemy of America, has some missiles, and all of its missiles were sold to it by Colonel West, who is American.

Prove that Col. West is a criminal

## Example knowledge base contd.

... it is a crime for an American to sell weapons to hostile nations:

## Example knowledge base contd.

... it is a crime for an American to sell weapons to hostile nations:

$$American(x) \wedge Weapon(y) \wedge Sells(x, y, z) \wedge Hostile(z) \Rightarrow Criminal(x)$$

Nono ... has some missiles

### Example knowledge base contd.

... it is a crime for an American to sell weapons to hostile nations:  
 $American(x) \wedge Weapon(y) \wedge Sells(x, y, z) \wedge Hostile(z) \Rightarrow Criminal(x)$   
 Nono ... has some missiles, i.e.,  $\exists x Owns(Nono, x) \wedge$   
 $Missile(x)$ :  $Owns(Nono, M_1)$  and  $Missile(M_1)$   
 ... all of its missiles were sold to it by Colonel West

### Example knowledge base contd.

... it is a crime for an American to sell weapons to hostile nations:  
 $American(x) \wedge Weapon(y) \wedge Sells(x, y, z) \wedge Hostile(z) \Rightarrow Criminal(x)$   
 Nono ... has some missiles, i.e.,  $\exists x Owns(Nono, x) \wedge$   
 $Missile(x)$ :  $Owns(Nono, M_1)$  and  $Missile(M_1)$   
 ... all of its missiles were sold to it by Colonel West  
 $\forall x Missile(x) \wedge Owns(Nono, x) \Rightarrow Sells(West, x, Nono)$   
 Missiles are weapons:

### Example knowledge base contd.

... it is a crime for an American to sell weapons to hostile nations:  
 $American(x) \wedge Weapon(y) \wedge Sells(x, y, z) \wedge Hostile(z) \Rightarrow Criminal(x)$   
 Nono ... has some missiles, i.e.,  $\exists x Owns(Nono, x) \wedge$   
 $Missile(x)$ :  $Owns(Nono, M_1)$  and  $Missile(M_1)$   
 ... all of its missiles were sold to it by Colonel West  
 $\forall x Missile(x) \wedge Owns(Nono, x) \Rightarrow Sells(West, x, Nono)$   
 Missiles are weapons:  
 $Missile(x) \Rightarrow Weapon(x)$   
 An enemy of America counts as "hostile":

### Example knowledge base contd.

... it is a crime for an American to sell weapons to hostile nations:  
 $American(x) \wedge Weapon(y) \wedge Sells(x, y, z) \wedge Hostile(z) \Rightarrow Criminal(x)$   
 Nono ... has some missiles, i.e.,  $\exists x Owns(Nono, x) \wedge$   
 $Missile(x)$ :  $Owns(Nono, M_1)$  and  $Missile(M_1)$   
 ... all of its missiles were sold to it by Colonel West  
 $\forall x Missile(x) \wedge Owns(Nono, x) \Rightarrow Sells(West, x, Nono)$   
 Missiles are weapons:  
 $Missile(x) \Rightarrow Weapon(x)$   
 An enemy of America counts as "hostile":  
 $Enemy(x, America) \Rightarrow Hostile(x)$   
 West, who is American ...  
 $American(West)$   
 The country Nono, an enemy of America ...  
 $Enemy(Nono, America)$

### Forward chaining algorithm

```
function FOL-FC-Ask(KB, a) returns a substitution or false
  repeat until new is empty
    new ← {}
    for each sentence r in KB do
       $(p_1 \wedge \dots \wedge p_n \Rightarrow q) \leftarrow \text{Standardize-Apart}(r)$ 
      for each  $\theta$  such that  $(p_1 \wedge \dots \wedge p_n)\theta =_I (p_1^1 \wedge \dots \wedge p_n^1)\theta$ 
        for some  $p_1^1, \dots, p_n^1$  in
           $q^1 \leftarrow \text{Subst}(\theta, q)$ 
          if  $q^1$  is not a renaming of a sentence already in KB or new then do
            add  $q^1$  to new
             $\varphi \leftarrow \text{Unify}(q^1, a)$ 
            if  $\varphi$  is not fail then return  $\varphi$ 
  add new to KB
  return false
```

### Forward chaining proof

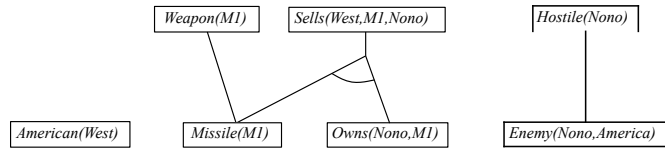
American(West)

Missile(M1)

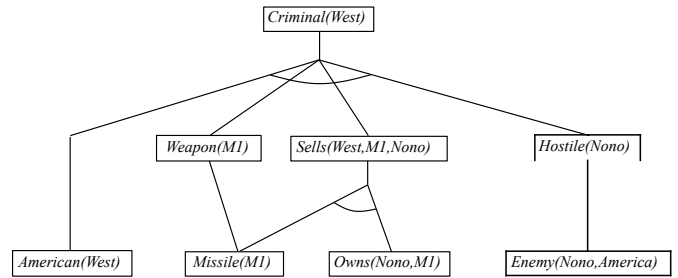
Owns(Nono, M1)

Enemy(Nono, America)

## Forward chaining proof



## Forward chaining proof



## Properties of forward chaining

Sound and complete for first-order definite clauses (proof similar to propositional proof)

**Datalog** = first-order definite clauses + **no functions** (e.g., crime KB)  
FC terminates for Datalog in poly iterations: at most  $p \cdot n^k$  literals

May not terminate in general if  $a$  is not entailed

This is unavoidable: entailment with definite clauses is semidecidable

## Efficiency of forward chaining

Simple observation: no need to match a rule on iteration  $k$   
if a premise wasn't added on iteration  $k - 1$   
 $\Rightarrow$  match each rule whose premise contains a newly added literal

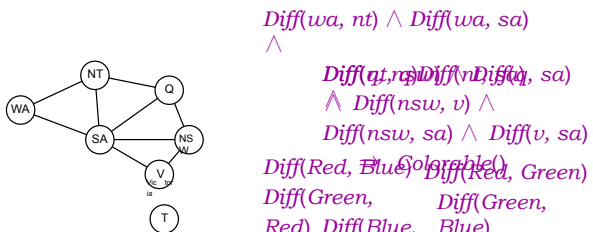
Matching itself can be expensive

**Database indexing** allows  $O(1)$  retrieval of known facts  
e.g., query  $Missile(x)$  retrieves  $Missile(M_1)$

Matching conjunctive premises against known facts is

NP-hard Forward chaining is widely used in **deductive databases**

## Hard matching example



**Colorable()** is inferred iff the CSP has a solution  
CSPs include 3SAT as a special case, hence matching is NP-hard

## Backward chaining algorithm

```

function FOL-BC-Ask(KB, goals,  $\theta$ ) returns a set of substitutions
inputs: KB, a knowledge base
goals, a list of conjuncts forming a query ( $\theta$  already applied)
 $\theta$ , the current substitution, initially the empty substitution {}
local variables: answers, a set of substitutions, initially empty
if goals is empty then return {}
 $q^1 \leftarrow \text{Subst}(\theta, \text{First}(\text{goals}))$ 
for each sentence  $r$  in KB
  where  $\text{Standardize-Apart}(r) = (p_1 \wedge \dots \wedge p_n \Rightarrow q)$ 
  and  $\theta' \leftarrow \text{Unify}(q, q^1)$  succeeds
  new_goals  $\leftarrow [p_1, \dots, p_n | \text{Rest}(\text{goals})]$ 
  answers  $\leftarrow \text{FOL-BC-Ask}(KB, \text{new\_goals}, \text{Compose}(\theta', \theta)) \cup \text{answers}$ 
return answers
  
```

## Backward chaining example

*Criminal(West)*

## Backward chaining example

*Criminal(West)*

$\{x/West\}$

*American(x)*

*Weapon(y)*

*Sells(x,y,z)*

*Hostile(z)*

## Backward chaining example

*Criminal(West)*

$\{x/West\}$

*American(West)*

$\{\}$

*Weapon(y)*

*Sells(x,y,z)*

*Hostile(z)*

## Backward chaining example

*Criminal(West)*

$\{x/West\}$

*American(West)*

$\{\}$

*Weapon(y)*

*Missile(y)*

*Sells(x,y,z)*

*Hostile(z)*

## Backward chaining example

*Criminal(West)*

$\{x/West, y/M1\}$

*American(West)*

$\{\}$

*Weapon(y)*

*Missile(y)*

$\{y/M1\}$

*Sells(x,y,z)*

*Hostile(z)*

## Backward chaining example

*Criminal(West)*

$\{x/West, y/M1, z/Nono\}$

*American(West)*

$\{\}$

*Weapon(y)*

*Missile(y)*

$\{y/M1\}$

*Sells(West,M1,z)*

$\{z/Nono\}$

*Missile(M1)*

*Owns(Nono,M1)*

*Hostile(z)*



## Conversion to CNF

Everyone who loves all animals is loved by

someone:

$$\forall x [\forall y \text{ Animal}(y) \Rightarrow \text{Loves}(x, y)] \Rightarrow [\exists y \text{ Loves}(y, x)]$$

1. Eliminate biconditionals and implications

$$\forall x [\neg \forall y \neg \text{Animal}(y) \vee \text{Loves}(x, y)] \vee [\exists y \text{ Loves}(y, x)]$$

2. Move  $\neg$  inwards:  $\neg \forall x p \equiv \exists x \neg p$ ,  $\neg \exists x p \equiv \forall x \neg p$

$$\forall x [\exists y \neg (\neg \text{Animal}(y) \vee \text{Loves}(x, y))] \vee [\exists y \text{ Loves}(y, x)]$$

$$\forall x [\exists y \neg \text{Loves}(y, x)] \vee [\exists y \text{ Loves}(y, x)]$$

$$\forall x [\exists y \neg \text{Animal}(y) \wedge \neg \text{Loves}(x, y)] \vee [\exists y \text{ Loves}(y, x)]$$

$$\forall x [\exists y \text{ Loves}(y, x)] \vee [\exists y \text{ Loves}(y, x)]$$

$$\forall x [\exists y \text{ Loves}(y, x)]$$

## Conversion to CNF contd.

3. Standardize variables: each quantifier should use a different one

$$\forall x [\exists y \text{ Animal}(y) \wedge \neg \text{Loves}(x, y)] \vee [\exists z \text{ Loves}(z, x)]$$

4. Skolemize: a more general form of existential instantiation. Each existential variable is replaced by a **Skolem function** of the enclosing universally quantified variables:

$$\forall x [\text{Animal}(F(x)) \wedge \neg \text{Loves}(x, F(x))] \vee \text{Loves}(G(x), x)$$

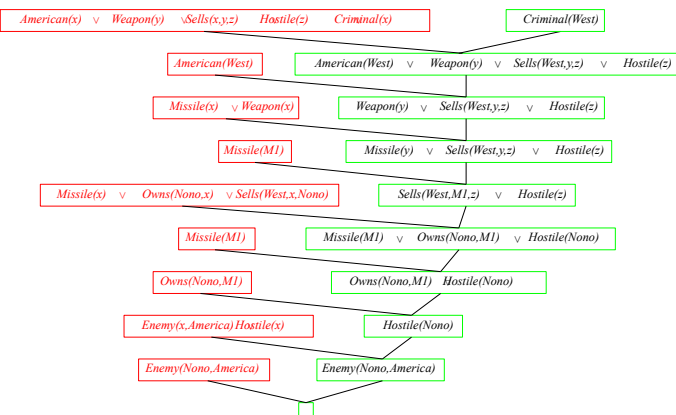
5. Drop universal quantifiers:

$$[\text{Animal}(F(x)) \wedge \neg \text{Loves}(x, F(x))] \vee \text{Loves}(G(x), x)$$

6. Distribute  $\wedge$  over  $\vee$ :

$$[\text{Animal}(F(x)) \vee \text{Loves}(G(x), x)] \wedge [\neg \text{Loves}(x, F(x)) \vee \text{Loves}(G(x), x)]$$

## Resolution proof: definite clauses



## Gödel's Incompleteness Theorem

- There are true arithmetic sentences that cannot be proved
- For any set of true sentences of number theory, and in particular any set of basic axioms, there are other true sentences that cannot be proved from those axioms.
- We can never prove all the theorems of mathematics within any given system of axioms.

## Resolution strategies

- Unit preference:** prefers to do resolutions where one of the sentences is a single literal (unit clause)
- Set of support:** every resolution step involve at least one element of a special set of clauses
- Input resolution:** every resolution combines one of the KB input sentences with other sentences
- Subsumption:** eliminates all sentences that are subsumed by KB sentences
- Learning:** learning from experience (machine learning)

## Summary

- Unification** identify appropriate substitutions for variables eliminates the instantiation step in first-order proofs, making the process more efficient in many cases
- Forward chaining** is used in deductive databases, where it can be combined with relational database operations. It is also used in production systems
- Backward chaining** is used in logic programming systems, which employ sophisticated compiler technology to provide very fast inference
- Prolog**, unlike first-order logic, uses a closed world with the unique names assumption and negation as failure.
- The generalized **resolution** inference rule provides a complete proof system for first order logic, using knowledge bases in conjunctive normal form.