

# Slides mainly from Poole & Mackworth, chap 13

**Datalog** based on the following assumptions

- An agent's knowledge can be usefully described in terms of *individuals* and *relations* among individuals.
- An agent's knowledge base consists of *definite* and *positive* statements.
- The environment is *static*.
- There are only a finite number of individuals of interest in the domain.  
An individual can be named.

# Datalog syntax

- A **variable** starts with upper-case letter.
- A **constant** starts with lower-case letter or is a sequence of digits (numeral).
- A **predicate symbol** starts with lower-case letter.
- A **term** is either a variable or a constant.
- An **atomic symbol** (atom) is of the form  $p$  or  $p(t_1, \dots, t_n)$  where  $p$  is a predicate symbol and  $t_i$  are terms.

## Datalog syntax (ctd)

- A **definite clause** is either an atomic symbol (a fact) or of the form:

$$\underbrace{a}_{\text{head}} \leftarrow \underbrace{b_1 \wedge \cdots \wedge b_m}_{\text{body}}$$

where  $a$  and  $b_i$  are atomic symbols.

- **query** is of the form  $?b_1 \wedge \cdots \wedge b_m$ .
- **knowledge base** is a set of definite clauses.

# Semantics: General Idea

A **semantics** specifies the meaning of sentences in the language.

An **interpretation** specifies:

- what objects (individuals) are in the world
- the correspondence between symbols in the computer and objects & relations in world
  - ▶ constants denote individuals
  - ▶ predicate symbols denote relations

# Formal Semantics

An **interpretation** is a triple  $I = \langle D, \phi, \pi \rangle$ , where

- $D$ , the **domain**, is a nonempty set. Elements of  $D$  are **individuals**.
- $\phi$  is a mapping that assigns to each constant an element of  $D$ . Constant  $c$  **denotes** individual  $\phi(c)$ .
- $\pi$  is a mapping that assigns to each  $n$ -ary predicate symbol a relation: a function from  $D^n$  into  $\{TRUE, FALSE\}$ .

# Example Interpretation

**Constants:** *phone*, *pencil*, *telephone*.

**Predicate Symbol:** *noisy* (unary), *left\_of* (binary).

- $D = \{\text{✂}, \text{☎}, \text{✎}\}.$
- $\phi(\text{phone}) = \text{☎}, \phi(\text{pencil}) = \text{✎}, \phi(\text{telephone}) = \text{☎}.$

- $\pi(\text{noisy})$ : 

$\langle \text{✂} \rangle$	FALSE	$\langle \text{☎} \rangle$	TRUE	$\langle \text{✎} \rangle$	FALSE
----------------------------	-------	----------------------------	------	----------------------------	-------

$\pi(\text{left\_of})$ :

$\langle \text{✂}, \text{✂} \rangle$	FALSE	$\langle \text{✂}, \text{☎} \rangle$	TRUE	$\langle \text{✂}, \text{✎} \rangle$	TRUE
$\langle \text{☎}, \text{✂} \rangle$	FALSE	$\langle \text{☎}, \text{☎} \rangle$	FALSE	$\langle \text{☎}, \text{✎} \rangle$	TRUE
$\langle \text{✎}, \text{✂} \rangle$	FALSE	$\langle \text{✎}, \text{☎} \rangle$	FALSE	$\langle \text{✎}, \text{✎} \rangle$	FALSE

## Important points to note

- The domain  $D$  can contain real objects. (e.g., a person, a room, a course).  $D$  can't necessarily be stored in a computer.
- $\pi(p)$  specifies whether the relation denoted by the  $n$ -ary predicate symbol  $p$  is true or false for each  $n$ -tuple of individuals.
- If predicate symbol  $p$  has no arguments, then  $\pi(p)$  is either *TRUE* or *FALSE*.

# Truth in an interpretation

A constant  $c$  **denotes in  $I$**  the individual  $\phi(c)$ .

Ground (variable-free) atom  $p(t_1, \dots, t_n)$  is

- **true in interpretation  $I$**  if  $\pi(p)(\langle \phi(t_1), \dots, \phi(t_n) \rangle) = \text{TRUE}$  in interpretation  $I$  and
- **false** otherwise.

Ground clause  $h \leftarrow b_1 \wedge \dots \wedge b_m$  is **false in interpretation  $I$**  if  $h$  is false in  $I$  and each  $b_i$  is true in  $I$ , and is **true in interpretation  $I$**  otherwise.



## Example Truths

In the interpretation given before, which of following are true?

*noisy(phone)*

*noisy(telephone)*

*noisy(pencil)*

*left\_of(phone, pencil)*

*left\_of(phone, telephone)*

*noisy(phone) ← left\_of(phone, telephone)*

*noisy(pencil) ← left\_of(phone, telephone)*

*noisy(pencil) ← left\_of(phone, pencil)*

*noisy(phone) ← noisy(telephone) ∧ noisy(pencil)*

## Example Truths

In the interpretation given before, which of following are true?

<i>noisy(phone)</i>	true
<i>noisy(telephone)</i>	true
<i>noisy(pencil)</i>	false
<i>left_of(phone, pencil)</i>	true
<i>left_of(phone, telephone)</i>	false
<i>noisy(phone) <math>\leftarrow</math> left_of(phone, telephone)</i>	true
<i>noisy(pencil) <math>\leftarrow</math> left_of(phone, telephone)</i>	true
<i>noisy(pencil) <math>\leftarrow</math> left_of(phone, pencil)</i>	false
<i>noisy(phone) <math>\leftarrow</math> noisy(telephone) <math>\wedge</math> noisy(pencil)</i>	true

# Models and logical consequences

- A knowledge base,  $KB$ , is true in interpretation  $I$  if and only if every clause in  $KB$  is true in  $I$ .
- A **model** of a set of clauses is an interpretation in which all the clauses are true.
- If  $KB$  is a set of clauses and  $g$  is a conjunction of atoms,  $g$  is a **logical consequence** of  $KB$ , written  $KB \models g$ , if  $g$  is true in every model of  $KB$ .
- That is,  $KB \models g$  if there is no interpretation in which  $KB$  is true and  $g$  is false.

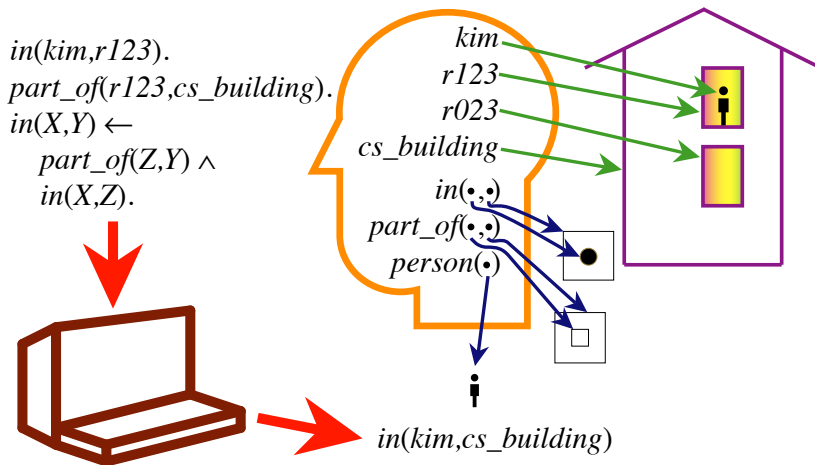
# User's view of Semantics

1. Choose a task domain: **intended interpretation**.
2. Associate constants with individuals you want to name.
3. For each relation you want to represent, associate a predicate symbol in the language.
4. Tell the system clauses that are true in the intended interpretation: **axiomatizing the domain**.
5. Ask questions about the intended interpretation.
6. If  $KB \models g$ , then  $g$  must be true in the intended interpretation.

# Computer's view of semantics

- The computer doesn't have access to the intended interpretation.
- All it knows is the knowledge base.
- The computer can determine if a formula is a logical consequence of KB.
- If  $KB \models g$  then  $g$  must be true in the intended interpretation.
- If  $KB \not\models g$  then there is a model of  $KB$  in which  $g$  is false.  
This could be the intended interpretation.

# Semantics in the mind



## Soundness and completeness

Recall that  $g$  is a *logical consequence* of  $KB$ ,  $KB \models g$ , precisely if  $g$  is true in all models of  $KB$ .

Let  $\vdash$  be a mechanical procedure for deriving a formula  $g$  from a knowledge base  $KB$ , written  $KB \vdash g$ .

$\vdash$  is **sound** if  $KB \models g$  whenever  $KB \vdash g$ .

$\vdash$  is **complete** if  $KB \vdash g$  whenever  $KB \models g$ .

# Soundness and completeness

Recall that  $g$  is a *logical consequence* of  $KB$ ,  $KB \models g$ , precisely if  $g$  is true in all models of  $KB$ .

Let  $\vdash$  be a mechanical procedure for deriving a formula  $g$  from a knowledge base  $KB$ , written  $KB \vdash g$ .

$\vdash$  is **sound** if  $KB \models g$  whenever  $KB \vdash g$ .

$\vdash$  is **complete** if  $KB \vdash g$  whenever  $KB \models g$ .

Two extreme examples:

(1)  $KB \vdash g$  for **no**  $g$   
sound

(2)  $KB \vdash g$  for **all**  $g$   
complete



# Propositional KBs

Recall

```
i :- p,q.  
i :- r.  
p.  
r.
```

# Propositional KBs

## Recall

```
i :- p,q.  
i :- r.  
p.  
r.
```

```
KB = [[i,p,q],[i,r],[p],[r]]  
arc([H|T],N,KB) :- member([H|B],KB),  
                    append(B,T,N).  
  
prove([],KB).  
prove(Node,KB) :- arc(Node,Next,KB),  
                  prove(Next,KB).
```

# Propositional KBs

Recall

$i :- p, q.$	$KB = [[i, p, q], [i, r], [p], [r]]$
$i :- r.$	$arc([H T], N, KB) :- member([H B], KB),$
$p.$	$append(B, T, N).$
$r.$	$prove([], KB).$
	$prove(Node, KB) :- arc(Node, Next, KB),$
	$prove(Next, KB).$

Let

$$KB \vdash G \iff prove([G], KB)$$

**Theorem.**

- (1)  $\vdash$  is sound (proved by induction)
- (2)  $\vdash$  is *not* complete (why?)

## Logical consequences bottom-up

$$C_0 := \emptyset$$

$$C_{n+1} := \{H \mid (\text{for some } B \subseteq C_n) \text{ member}([H|B], KB)\}$$

$$C := \bigcup_{n \geq 0} C_n$$

$$= \bigcup_{n \leq k} C_n \quad \text{where } k = \text{number of clauses in } KB$$

## Logical consequences bottom-up

$$\begin{aligned}C_0 &:= \emptyset \\C_{n+1} &:= \{H \mid (\text{for some } B \subseteq C_n) \text{ member}([H|B], KB)\} \\C &:= \bigcup_{n \geq 0} C_n \\&= \bigcup_{n \leq k} C_n \quad \text{where } k = \text{number of clauses in } KB\end{aligned}$$

i :- p,q.	KB = [[i,p,q],[i,r],[p],[r]]
i :- r.	arc([H T],N,KB) :- member([H B],KB),
p.	append(B,T,N).
r.	$C_1 = \{p,r\}$
	$C_2 = \{p,r,i\} = C_n \text{ for } n \geq 2$

A 0-ary predicate  $p$  is interpreted by  $I = \langle D, \phi, \pi \rangle$  as

$$\pi(p) : D^0 \rightarrow \{\text{true}, \text{false}\}.$$

## Substitutions and instances

A 0-ary predicate  $p$  is interpreted by  $I = \langle D, \phi, \pi \rangle$  as

$$\pi(p) : D^0 \rightarrow \{\text{true}, \text{false}\}.$$

Let  $K$  be a set of constants.

A  **$K$ -substitution** is a function from a finite set of variables to  $K$  — i.e. a set  $\{V_1/c_1, \dots, V_n/c_n\}$  of  $c_i \in K$  and distinct variables  $V_i$ .

The **application  $e\theta$**  of a  $K$ -substitution  $\theta = \{V_1/c_1, \dots, V_n/c_n\}$  to a clause  $e$  is  $e$  with each  $V_i$  replaced by  $c_i$

$$\text{e.g. } p(Z, U, Y, a, X)\{X/b, U/a, Z/b\} = p(b, a, Y, a, b).$$

A  **$K$ -instance** of  $e$  is  $e\theta$  for some  $K$ -substitution  $\theta$ .

Given a set  $B$  of clauses and a  $K$ -substitution  $\theta$ , let

$$B\theta := \{e\theta \mid e \in B\}.$$

## Bottom-up with substitutions

If  $KB$  has constants from some non-empty finite set  $K$ , let

$$C_0^K := \emptyset$$

$$C_{n+1}^K := \{H\theta \mid \theta \text{ is a } K\text{-substitution s.t. } B\theta \subseteq C_n \\ \text{for some } B \text{ s.t. } \text{member}([H|B], KB)\}$$

$$C^K := \bigcup_{n \geq 0} C_n^K$$



## Bottom-up with substitutions

If  $KB$  has constants from some non-empty finite set  $K$ , let

$$C_0^K := \emptyset$$

$$C_{n+1}^K := \{H\theta \mid \theta \text{ is a } K\text{-substitution s.t. } B\theta \subseteq C_n \\ \text{for some } B \text{ s.t. } \text{member}([H|B], KB)\}$$

$$C^K := \bigcup_{n \geq 0} C_n^K$$

E.g. for  $KB = [[p(a, b)], [q(X), p(X, Y)]]$  and  $K = \{a, b\}$ ,

$$C_1^K = \{p(a, b)\}$$

$$C_2^K = \{p(a, b), q(a)\} = C^K$$

# Soundness & completeness via Herbrand

The **Herbrand interpretation** of a set  $KB$  of clauses with constants from a non-empty set  $K$  is the triple  $I = \langle D, \phi, \pi \rangle$  where

- the domain  $D$  is the set  $K$  of constants
- $\phi$  is the identity function on  $K$  (each constant in  $K$  refers to itself)
- for each  $n$ -ary  $p$  and  $n$ -tuple  $c_1 \dots c_n$  from  $K$ ,

$$\pi(p)(c_1 \dots c_n) = \text{true} \iff p(c_1 \dots c_n) \in C^K$$

# Soundness & completeness via Herbrand

The **Herbrand interpretation** of a set  $KB$  of clauses with constants from a non-empty set  $K$  is the triple  $I = \langle D, \phi, \pi \rangle$  where

- the domain  $D$  is the set  $K$  of constants
- $\phi$  is the identity function on  $K$  (each constant in  $K$  refers to itself)
- for each  $n$ -ary  $p$  and  $n$ -tuple  $c_1 \dots c_n$  from  $K$ ,

$$\pi(p)(c_1 \dots c_n) = \text{true} \iff p(c_1 \dots c_n) \in C^K$$

**Fact.**  $I$  is a model of  $KB$ , and every clause true in  $I$  is true in every model of  $KB$  (interpreting constants in  $K$ ).

# Soundness & completeness via Herbrand

The **Herbrand interpretation** of a set  $KB$  of clauses with constants from a non-empty set  $K$  is the triple  $I = \langle D, \phi, \pi \rangle$  where

- the domain  $D$  is the set  $K$  of constants
- $\phi$  is the identity function on  $K$  (each constant in  $K$  refers to itself)
- for each  $n$ -ary  $p$  and  $n$ -tuple  $c_1 \dots c_n$  from  $K$ ,

$$\pi(p)(c_1 \dots c_n) = \text{true} \iff p(c_1 \dots c_n) \in C^K$$

**Fact.**  $I$  is a model of  $KB$ , and every clause true in  $I$  is true in every model of  $KB$  (interpreting constants in  $K$ ).

**Corollary.** The bottom-up procedure with substitutions is sound and complete (for Datalog).