

# Complete Knowledge Assumption

- Often you want to assume that your knowledge is complete.
- **Example:** assume that a database of what students are enrolled in a course is complete. We don't want to have to state all negative enrolment facts!
- The definite clause language is **monotonic**: adding clauses can't invalidate a previous conclusion.
- Under the complete knowledge assumption, the system is **non-monotonic**: adding clauses can invalidate a previous conclusion.

# Equality

Equality is a special predicate symbol with a standard domain-independent intended interpretation.

- Suppose interpretation  $I = \langle D, \phi, \pi \rangle$ .
- $t_1$  and  $t_2$  are ground terms then  $t_1 = t_2$  is true in interpretation  $I$  if  $t_1$  and  $t_2$  denote the same individual. That is,  $t_1 = t_2$  if  $\phi(t_1)$  is the same as  $\phi(t_2)$ .
- $t_1 \neq t_2$  when  $t_1$  and  $t_2$  denote different individuals.

# Equality

Equality is a special predicate symbol with a standard domain-independent intended interpretation.

- Suppose interpretation  $I = \langle D, \phi, \pi \rangle$ .
- $t_1$  and  $t_2$  are ground terms then  $t_1 = t_2$  is true in interpretation  $I$  if  $t_1$  and  $t_2$  denote the same individual. That is,  $t_1 = t_2$  if  $\phi(t_1)$  is the same as  $\phi(t_2)$ .
- $t_1 \neq t_2$  when  $t_1$  and  $t_2$  denote different individuals.
- Example:

$D = \{ \text{✂}, \text{☎}, \text{✎} \}$ .

$\phi(\text{phone}) = \text{☎}$ ,  $\phi(\text{pencil}) = \text{✎}$ ,  $\phi(\text{telephone}) = \text{☎}$

What equalities and inequalities hold?

Equality is a special predicate symbol with a standard domain-independent intended interpretation.

- Suppose interpretation  $I = \langle D, \phi, \pi \rangle$ .
- $t_1$  and  $t_2$  are ground terms then  $t_1 = t_2$  is true in interpretation  $I$  if  $t_1$  and  $t_2$  denote the same individual. That is,  $t_1 = t_2$  if  $\phi(t_1)$  is the same as  $\phi(t_2)$ .
- $t_1 \neq t_2$  when  $t_1$  and  $t_2$  denote different individuals.
- Example:

$D = \{ \text{✂}, \text{☎}, \text{✎} \}$ .

$\phi(\text{phone}) = \text{☎}$ ,  $\phi(\text{pencil}) = \text{✎}$ ,  $\phi(\text{telephone}) = \text{☎}$

What equalities and inequalities hold?

$\text{phone} = \text{telephone}$ ,  $\text{phone} = \text{phone}$ ,  $\text{pencil} = \text{pencil}$ ,

$\text{telephone} = \text{telephone}$

$\text{pencil} \neq \text{phone}$ ,  $\text{pencil} \neq \text{telephone}$

# Equality

Equality is a special predicate symbol with a standard domain-independent intended interpretation.

- Suppose interpretation  $I = \langle D, \phi, \pi \rangle$ .
- $t_1$  and  $t_2$  are ground terms then  $t_1 = t_2$  is true in interpretation  $I$  if  $t_1$  and  $t_2$  denote the same individual. That is,  $t_1 = t_2$  if  $\phi(t_1)$  is the same as  $\phi(t_2)$ .
- $t_1 \neq t_2$  when  $t_1$  and  $t_2$  denote different individuals.

- Example:

$D = \{ \text{✂}, \text{☎}, \text{✎} \}$ .

$\phi(\text{phone}) = \text{☎}$ ,  $\phi(\text{pencil}) = \text{✎}$ ,  $\phi(\text{telephone}) = \text{☎}$

What equalities and inequalities hold?

$\text{phone} = \text{telephone}$ ,  $\text{phone} = \text{phone}$ ,  $\text{pencil} = \text{pencil}$ ,

$\text{telephone} = \text{telephone}$

$\text{pencil} \neq \text{phone}$ ,  $\text{pencil} \neq \text{telephone}$

- Equality does not mean similarity!

# Properties of Equality

Equality is:

- Reflexive:  $X = X$
- Symmetric: if  $X = Y$  then  $Y = X$
- Transitive: if  $X = Y$  and  $Y = Z$  then  $X = Z$

For each  $n$ -ary function symbol  $f$

$$f(X_1, \dots, X_n) = f(Y_1, \dots, Y_n) \text{ if } X_1 = Y_1 \text{ and } \dots \text{ and } X_n = Y_n.$$

For each  $n$ -ary predicate symbol  $p$

$$p(X_1, \dots, X_n) \text{ if } p(Y_1, \dots, Y_n) \text{ and } X_1 = Y_1 \text{ and } \dots \text{ and } X_n = Y_n.$$

# Unique Names Assumption

- Suppose the only clauses for *enrolled* are

*enrolled(sam, cs222)*

*enrolled(chris, cs222)*

*enrolled(sam, cs873)*

To conclude  $\neg \textit{enrolled}(\textit{chris}, \textit{cs873})$ , what do we need to assume?

# Unique Names Assumption

- Suppose the only clauses for *enrolled* are

*enrolled(sam, cs222)*

*enrolled(chris, cs222)*

*enrolled(sam, cs873)*

To conclude  $\neg \textit{enrolled}(\textit{chris}, \textit{cs873})$ , what do we need to assume?

- ▶ All other enrolled facts are false
- ▶ Inequalities:

$$\textit{sam} \neq \textit{chris} \wedge \textit{cs873} \neq \textit{cs222}$$

- The **unique names assumption (UNA)** is the assumption that distinct ground terms denote different individuals.



# Completion of a knowledge base: propositional case

- Suppose the rules for atom  $a$  are

$$a \leftarrow b_1.$$

$\vdots$

$$a \leftarrow b_n.$$

equivalently  $a \leftarrow b_1 \vee \dots \vee b_n.$

- Under the Complete Knowledge Assumption, if  $a$  is true, one of the  $b_i$  must be true:

$$a \rightarrow b_1 \vee \dots \vee b_n.$$

- Thus, the clauses for  $a$  mean

$$a \leftrightarrow b_1 \vee \dots \vee b_n$$

# Clark Normal Form

The **Clark normal form** of the clause

$$p(t_1, \dots, t_k) \leftarrow B.$$

is the clause

$$p(V_1, \dots, V_k) \leftarrow \exists W_1 \dots \exists W_m V_1 = t_1 \wedge \dots \wedge V_k = t_k \wedge B.$$

where

- $V_1, \dots, V_k$  are  $k$  variables that did not appear in the original clause
- $W_1, \dots, W_m$  are the original variables in the clause.

# Clark Normal Form

The **Clark normal form** of the clause

$$p(t_1, \dots, t_k) \leftarrow B.$$

is the clause

$$p(V_1, \dots, V_k) \leftarrow \exists W_1 \dots \exists W_m V_1 = t_1 \wedge \dots \wedge V_k = t_k \wedge B.$$

where

- $V_1, \dots, V_k$  are  $k$  variables that did not appear in the original clause
- $W_1, \dots, W_m$  are the original variables in the clause.
- When the clause is an atomic clause,  $B$  is *true*.
- Often can be simplified by replacing  $\exists W V = W \wedge p(W)$  with  $P(V)$ .

# Clark normal form

For the clauses

$student(mary).$

$student(sam).$

$student(X) \leftarrow undergrad(X).$

the Clark normal form is

$student(V) \leftarrow V = mary.$

$student(V) \leftarrow V = sam.$

$student(V) \leftarrow \exists X V = X \wedge undergrad(X).$

# Clark's Completion

Suppose all of the clauses for  $p$  are put into Clark normal form, with the same set of introduced variables, giving

$$p(V_1, \dots, V_k) \leftarrow B_1.$$

$\vdots$

$$p(V_1, \dots, V_k) \leftarrow B_n.$$

which is equivalent to

$$p(V_1, \dots, V_k) \leftarrow B_1 \vee \dots \vee B_n.$$

**Clark's completion** of predicate  $p$  is the equivalence

$$\forall V_1 \dots \forall V_k p(V_1, \dots, V_k) \leftrightarrow B_1 \vee \dots \vee B_n$$

If there are no clauses for  $p$ ,

# Clark's Completion

Suppose all of the clauses for  $p$  are put into Clark normal form, with the same set of introduced variables, giving

$$p(V_1, \dots, V_k) \leftarrow B_1.$$

$\vdots$

$$p(V_1, \dots, V_k) \leftarrow B_n.$$

which is equivalent to

$$p(V_1, \dots, V_k) \leftarrow B_1 \vee \dots \vee B_n.$$

**Clark's completion** of predicate  $p$  is the equivalence

$$\forall V_1 \dots \forall V_k p(V_1, \dots, V_k) \leftrightarrow B_1 \vee \dots \vee B_n$$

If there are no clauses for  $p$ , the completion results in

$$\forall V_1 \dots \forall V_k p(V_1, \dots, V_k) \leftrightarrow \textit{false}$$

**Clark's completion** of a knowledge base consists of the completion of every predicate symbol along the unique names assumption.

# Completion example

$$p \leftarrow q \wedge \sim r.$$

$$p \leftarrow s.$$

$$q \leftarrow \sim s.$$

$$r \leftarrow \sim t.$$

$$t.$$

$$s \leftarrow w.$$

## Completion Example

Consider the recursive definition:

$passed\_each([], St, MinPass).$

$passed\_each([C|R], St, MinPass) \leftarrow$   
 $passed(St, C, MinPass) \wedge$   
 $passed\_each(R, St, MinPass).$

In Clark normal form, this can be written as



## Completion Example

Consider the recursive definition:

$$\begin{aligned} & \textit{passed\_each}([], St, MinPass). \\ & \textit{passed\_each}([C|R], St, MinPass) \leftarrow \\ & \quad \textit{passed}(St, C, MinPass) \wedge \\ & \quad \textit{passed\_each}(R, St, MinPass). \end{aligned}$$

In Clark normal form, this can be written as

$$\begin{aligned} & \textit{passed\_each}(L, S, M) \leftarrow L = []. \\ & \textit{passed\_each}(L, S, M) \leftarrow \\ & \quad \exists C \exists R L = [C|R] \wedge \textit{passed}(S, C, M) \wedge \textit{passed\_each}(R, S, M). \end{aligned}$$

Here we renamed the variables as appropriate. Thus, Clark's completion of *passed\_each* is

# Completion Example

Consider the recursive definition:

$$\begin{aligned} & \textit{passed\_each}([], St, MinPass). \\ & \textit{passed\_each}([C|R], St, MinPass) \leftarrow \\ & \quad \textit{passed}(St, C, MinPass) \wedge \\ & \quad \textit{passed\_each}(R, St, MinPass). \end{aligned}$$

In Clark normal form, this can be written as

$$\begin{aligned} & \textit{passed\_each}(L, S, M) \leftarrow L = []. \\ & \textit{passed\_each}(L, S, M) \leftarrow \\ & \quad \exists C \exists R L = [C|R] \wedge \textit{passed}(S, C, M) \wedge \textit{passed\_each}(R, S, M). \end{aligned}$$

Here we renamed the variables as appropriate. Thus, Clark's completion of *passed\_each* is

$$\begin{aligned} & \forall L \forall S \forall M \textit{passed\_each}(L, S, M) \leftrightarrow L = [] \vee \\ & \quad \exists C \exists R L = [C|R] \wedge \textit{passed}(S, C, M) \wedge \textit{passed\_each}(R, S, M). \end{aligned}$$

# Clark's Completion of a KB

- Clark's completion of a knowledge base consists of the completion of every predicate.
- The completion of an  $n$ -ary predicate  $p$  with no clauses is  $p(V_1, \dots, V_n) \leftrightarrow \text{false}$ .
- You can interpret negations in the body of clauses.  $\sim a$  means  $a$  is false under the complete knowledge assumption.  $\sim a$  is replaced by  $\neg a$  in the completion. This is **negation as failure**.

## Defining *empty\_course*

Given database of:

- *course*( $C$ ) that is true if  $C$  is a course
- *enrolled*( $S, C$ ) that is true if student  $S$  is enrolled in course  $C$ .

Define *empty\_course*( $C$ ) that is true if there are no students enrolled in course  $C$ .

## Defining *empty\_course*

Given database of:

- $course(C)$  that is true if  $C$  is a course
- $enrolled(S, C)$  that is true if student  $S$  is enrolled in course  $C$ .

Define  $empty\_course(C)$  that is true if there are no students enrolled in course  $C$ .

- Using negation as failure,  $empty\_course(C)$  can be defined by
$$empty\_course(C) \leftarrow course(C) \wedge \sim has\_enrollment(C).$$
$$has\_enrollment(C) \leftarrow enrolled(S, C).$$

## Defining *empty\_course*

Given database of:

- $course(C)$  that is true if  $C$  is a course
- $enrolled(S, C)$  that is true if student  $S$  is enrolled in course  $C$ .

Define  $empty\_course(C)$  that is true if there are no students enrolled in course  $C$ .

- Using negation as failure,  $empty\_course(C)$  can be defined by
$$empty\_course(C) \leftarrow course(C) \wedge \sim has\_enrollment(C).$$
$$has\_enrollment(C) \leftarrow enrolled(S, C).$$
- The completion of this is:

## Defining *empty\_course*

Given database of:

- *course*(*C*) that is true if *C* is a course
- *enrolled*(*S*, *C*) that is true if student *S* is enrolled in course *C*.

Define *empty\_course*(*C*) that is true if there are no students enrolled in course *C*.

- Using negation as failure, *empty\_course*(*C*) can be defined by
$$\begin{aligned} \text{empty\_course}(C) &\leftarrow \text{course}(C) \wedge \sim \text{has\_enrollment}(C). \\ \text{has\_enrollment}(C) &\leftarrow \text{enrolled}(S, C). \end{aligned}$$

- The completion of this is:

$$\begin{aligned} \forall C \text{ empty\_course}(C) &\iff \text{course}(C) \wedge \neg \text{has\_enrollment}(C). \\ \forall C \text{ has\_enrollment}(C) &\iff \exists S \text{ enrolled}(S, C). \end{aligned}$$

# Bottom-up negation as failure interpreter

```
C := {};  
repeat  
  either  
    select  $r \in KB$  such that  
       $r$  is " $h \leftarrow b_1 \wedge \dots \wedge b_m$ "  
       $b_i \in C$  for all  $i$ , and  
       $h \notin C$ ;  
     $C := C \cup \{h\}$   
  or  
    select  $h$  such that for every rule " $h \leftarrow b_1 \wedge \dots \wedge b_m$ "  $\in KB$   
      either for some  $b_i, \sim b_i \in C$   
      or some  $b_i = \sim g$  and  $g \in C$   
     $C := C \cup \{\sim h\}$   
until no more selections are possible
```



# Negation as failure example

$p \leftarrow q \wedge \sim r.$

$p \leftarrow s.$

$q \leftarrow \sim s.$

$r \leftarrow \sim t.$

$t.$

$s \leftarrow w.$

# Top-Down negation as failure proof procedure

- If the proof for  $a$  fails, you can conclude  $\sim a$ .
- Failure can be defined recursively:  
Suppose you have rules for atom  $a$ :

$$a \leftarrow b_1$$

$$\vdots$$

$$a \leftarrow b_n$$

If each body  $b_i$  fails,  $a$  fails.

- A body fails if one of the conjuncts in the body fails.
- Note that you need *finite* failure. Example  $p \leftarrow p$ .

$p(X) \leftarrow \sim q(X) \wedge r(X).$

$q(a).$

$q(b).$

$r(d).$

ask  $p(X).$

- What is the answer to the query?

$p(X) \leftarrow \sim q(X) \wedge r(X).$

$q(a).$

$q(b).$

$r(d).$

ask  $p(X).$

- What is the answer to the query?
- How can a top-down proof procedure find the answer?

$p(X) \leftarrow \sim q(X) \wedge r(X).$

$q(a).$

$q(b).$

$r(d).$

ask  $p(X).$

- What is the answer to the query?
- How can a top-down proof procedure find the answer?
- Delay the subgoal until it is bound enough.  
Sometimes it never gets bound enough — “floundering”.

# Problematic Cases

$p(X) \leftarrow \sim q(X)$

$q(X) \leftarrow \sim r(X)$

$r(a)$

ask  $p(X)$ .

- What is the answer?

# Problematic Cases

$p(X) \leftarrow \sim q(X)$

$q(X) \leftarrow \sim r(X)$

$r(a)$

ask  $p(X)$ .

- What is the answer?
- What does delaying do?

# Problematic Cases

$p(X) \leftarrow \sim q(X)$

$q(X) \leftarrow \sim r(X)$

$r(a)$

ask  $p(X)$ .

- What is the answer?
- What does delaying do?
- How can this be implemented?