LECTURE 7
Logic Programming

Note if Language $\mathcal{L}$ is not specified a priori then:

- by default it is <u>inherited</u> from the language used in PROLOG program.

Given $\mathcal{L}$ containing open or/and closed formulas there are <u>many</u> <u>different</u> <u>interpretations</u>:

It suffices to vary either:

a) D

b) or/and $a_i \longrightarrow a_i' \in D$ or

c) $f/n \longrightarrow f': \mathbb{D}^n \longrightarrow D$ or

d) $p/n \longrightarrow p': \mathbb{D}^n \longrightarrow \{true, false\}$

to define each time different interpretation.

LECTURE 7
LOGIC PROGRAMMING

Therefore under different interpretations:

for $f \in \Delta \Rightarrow$ **$I(f)$ can be true or false**

This was the same for PROPOSITIONAL LOGIC – see truth table.

Example 1: $\mathcal{L} = \{a; p/1\}$.

(i) Let $D = [0, +\infty)$ $\quad a \to 0$

$p \to p'$ is true if $x \geq 0$.

For $\Delta = \{\forall x\, p(x)\}$

f

under $I_{(i)}$ interpretation $I_{(i)}(f) = \text{true}$.

(ii) Let $D = [1, +\infty)$ the rest the same

under $I_{(ii)}(f) = \text{false}$.

□

So we generalize now some notions introduced in PROPOSITIONAL LOGIC.

LECTURE 7
LOGIC PROGRAMMING

DEFINITION 1:

Let S be a set of <u>closed formulas</u> from PREDICATE LOGIC.
Then an interpretation $\mathcal{I}$ is <u>a model</u> for S if all formulas in S are true under $\mathcal{I}$:

i.e. $\boxed{\forall f \in S \Rightarrow \mathcal{I}(f) = \text{true}.}$

DEFINITION 2:

Let S be a set of closed formulas from predicate logic.
Then S is <u>satisfiable</u> if there is some interpretation $\mathcal{I}$ that is a model for S. S <u>is unsatisfiable</u> if there is model for S.
If every interpretation I is a model for S then S is called VALID.
(extention of TAUTOLOGY in PROPOSITIONAL LOGIC

LECTURE 7
LOGIC PROGRAMMING

Example 2:

Consider the formula $\exists x\, p(x)$. (*)

Truth value depends on interpretation.

a) $D = \{0\}$ $\quad p'(0) = T$

$\exists x\, p(x)$ is true

so I_a is a model

b) $D = \{0\}$ $\quad p'(0) = F$

$\exists x\, p(x)$ is false

so I_b is not a model

(*) is <u>satisfiable</u> but is <u>not valid</u>. □

Example 3:

$$\forall x\, p(x) \leftrightarrow \exists x\, p(x). \quad (**)$$

"$\leftrightarrow$" (L R) true if $R \Leftarrow L$ & $L \Leftarrow R$

a) D any set $\quad p'(x)$ is true for any $x \in D$.

$\forall x\, p(x)$ is true $\Leftrightarrow$ $\exists x\, p(x)$ is true for I_a.

So (**) is true under $\mathcal{I}_a$.

(b) $D = \{0, 1\}$ $\quad p'(0) = \text{true}$
$\quad p'(1) = \text{false}$

I_b is not a model for (**) □

Example 4:

(*) $\underbrace{\exists x\, p(x,x)}_{\text{Head}} \leftarrow \underbrace{\forall x \forall y \exists z (p(x,z) \wedge p(z,y))}_{\text{Body}}$

For (*) to be INVALID one has to find one interpretation which is not a model for (*).

If one cannot find a counterexample then one can try to prove the validity.

For $\mathcal{I}$ to be not a model for (*) we need

$$\underbrace{\mathcal{I}(\text{Head})}_{\text{false}} \leftarrow \underbrace{\mathcal{I}(\text{Body})}_{\text{true}}$$

Take

a) D arbitrary

b) $p' \equiv ' \neq '$

$\Rightarrow$ $\forall x \forall y \exists z (x \neq z \wedge y \neq z)$ is true.

$\exists x \; x \neq x$ is false.

Thus (*) is false.

□

REMARK: In both examples above, the Interpretations for which $I(f) =$ false were found specifically.

QUESTION:

How to think of a given formula without <u>testing</u> their <u>satisfiability</u> for specific interpretations.

LECTURE 7
LOGIC PROGRAMMING

There are few tricks!

(a) for <u>unary predicate</u> P/1:

Each interpretation can be regarded as a subest of $D' \subset D$.

Given D: for a subset D' we put

$$\begin{cases} \text{if } d \in D' \Rightarrow p'(d) = \text{true} \\ \text{if } d \notin D' \Rightarrow p'(d) = \text{false} \end{cases}$$

This recalls the situation from PROPOSITIONAL LOGIC (A)

$\Delta = \{f_1, \ldots, f_n\} \rightarrow \Delta_a = \{p_1, \ldots, p_m\}$

Each interpretation in (A) is a subset of Δ_a i.e.

$D' \subset \Delta_a \Rightarrow$

$p_i \in D' \Rightarrow I(p_i) = \text{true}$

$p_i \notin D' \Rightarrow I(p_i) = \text{false}$

(b) for <u>binary predicate</u> p/2:

For a given D we define

a graph

with

(i) vertices from D

(ii) edges (directed ones)

$g_1, g_2 \in D$

there is an edge from g_1 to g_2 if $\boxed{p'(g_1, g_2) = true}$

Thus in our previous examples:

LECTURE 7
LOGIC PROGRAMMING

- $\forall x \forall y \exists z (p(x,z) \wedge p(z,y))$

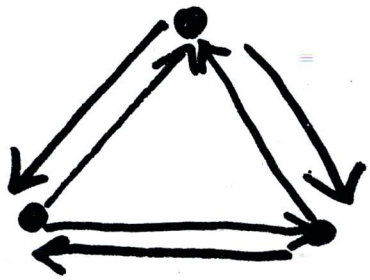
"there should (for counterexample) be a path of length 2 between any two vertices".

- $\exists x\, p(x,x)$

"there is a loop for some vertex."

It is easy to find a graph satisfying ■ & not satisfying ■

i.e.



Thus from such a graph we can read an interpretation which disproves (*) from Example 4.

LECTURE 7
LOGIC PROGRAMMING

$G = \{g_1, g_2, g_3\}$

$p \rightarrow p'$:

true= $P'(g_1, g_2) = P'(g_1, g_3) = P'(g_2, g_3)$

true= $P'(g_2, g_1) = P'(g_3, g_1) = P'(g_3, g_2)$

$P'(g_1, g_1) = P'(g_2, g_2) = P'(g_3, g_3)$

$= \text{false}$

Geometrical approach can help in deriving the proof for

INVALIDITY

of a given formula or set of formulas from Δ.

If we cannot derive such counterexample

PROOF for VALIDITY <u>is needed</u>.

LECTURE 7
LOGIC PROGRAMMING

DEFINITION 3:

Let S be a set of closed formulas.
Then a closed formula F is

a logical consequence

of S, denoted by

$$S \models F$$

if each model for S is a model for F.

Note the definition of $S \models F$ is analogous to this one for $\Delta \models F$

PROPOSITIONAL LOGIC.

The difference is:

the notion of a model is used for PREDICATE LOGIC in more general sense.

LECTURE 4
LOGIC PROGRAMMING

Example 5:

$$\blacksquare \begin{cases} \forall x\,(p(x) \leftarrow q(x)) \equiv p(x) \leftarrow q(x). \\ q(a). \end{cases}$$

QUESTION: $S \models^{?} p(a)$

PROOF:

Take arbitrary model for $\blacksquare$.

Thus as both clauses are to be true under $\mathcal{I}$ the 1st clause is true

$\blacksquare_1$ $\forall x\,(p(x) \leftarrow q(x))$. is true.

Therefore a GROUND INSTANCE of $\blacksquare_1$ is true for $x=a$, i.e.

$$p(a) \leftarrow q(a)$$

as $q(a)$ is true (i.e. $\blacksquare_2$) so applying RESOLUTION to ground instances:

LECTURE 7

LOGIC PROGRAMMING

$$\left.\begin{array}{l} p(a) \leftarrow q(a) \\ q(a) \end{array}\right\} \text{RESOLUTION}^{*}$$

$$\overline{\qquad p(a). \qquad}$$ RESOLVENT

It is easy to show by using exactly the same technique (proof) that RESOLUTION is SOUND for ground clauses.

Thus $\boxed{\blacksquare \models p(a).}$

□

SLD - RESOLUTION for PREDICATE LOGIC is used in LOGIC PROGRAMMING - see later.

* Note: we use RESOLUTION here in more general case than for PROPOSITIONAL LOGIC (i.e. for GROUND INSTANCES). Further extension is used later.

An easy the same proof as for PROPOSITIONAL LOGIC shows:

Th 1: Let S be a set of closed formulas. Then F is a **logical consequence** of S iff $S \cup \{\sim F\}$ is **unsatisfiable**.

Again the hard theorem (as for PROPOSITIONAL LOGIC) is also here (true) to be proved (see later):

Th 2: S is **unsatisfiable** iff $\square \in (S \cup \{\sim F\})^*$.

Let us return to Example 5 for which we use now:

REFUTATION

LECTURE 7
LOGIC PROGRAMMING

Example 6:

$$S = \{\ p(x) \leftarrow q(x)\ ;\ q(a)\ \}$$

$$\boxed{S \models p(a)} \quad ?$$

- first **negate** $p(a)$ & **add** it to S:

$$\tilde{S} = S \cup \{ \sim p(a) \}$$
$$\| \quad\quad \| \quad\quad \| \quad\quad \|$$
$$\leftarrow p(a).$$

- then take a **ground instance** of $p(x) \leftarrow q(x)$, namely

$$p(a) \leftarrow q(a).$$

- resolve $p(a) \leftarrow q(a)$ with $\leftarrow p(a)$

$$\frac{p(a) \leftarrow q(a) \qquad \leftarrow p(a)}{\leftarrow q(a)}$$ RESOLVENT 1.

- resolve $\leftarrow q(a)$ with $q(a)$

$$\frac{\leftarrow q(a) \qquad q(a)}{\square}$$

by Th 1 & Th 2 $\boxed{S \models p(a)}$ $\square$

LECTURE 7
LOGIC PROGRAMMING

We can now come back to the example with Confucius & his mortality.

Example 7:

$$\Delta \begin{cases} \forall x\,(mortal(x) \leftarrow man(x)). \\ man(confucius). \end{cases}$$

QUESTION:

$$\Delta \stackrel{?}{\models} mortal(confucius).$$

Take **ground instance** of Δ_1

$$mortal(confucius) \leftarrow man(confucius)$$

& resolve it with Δ_2 i.e.

$$\frac{mortal(confucius) \leftarrow man(confucius). \qquad man(confucius).}{mortal(confucius).}$$

RESOLVENT

LECTURE 7
LOGIC PROGRAMMING

Taking into account that

RESOLUTION IS SOUND*

we get

$\Delta \vdash mortal(confucius)$.

Alternatively this can also be shown by REFUTATION

↑

see previous example

□

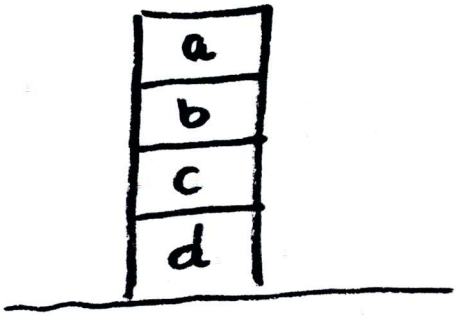
(*) to be proved soon for more general setting than PROPOSITIONAL LOGIC OR ground instances in PREDICATE LOGIC

LECTURE 7

LOGIC PROGRAMMING.

Example 8:

Let "world" consists of objects a, b, c, & d.



The relationship:

"positioned over"

over/2

over/2 is transitive can be expressed through a purely

extentional database

```
over(a,b).    over(b,c).
over(a,c).    over(b,d).
over(a,d).    over(c,d).
```

LECTURE 7

LOGIC PROGRAMMING

Alternatively it can be expressed as:

deductive database

over(X,Z) ← over(X,Y), over(Y,Z).

■ over(a,b).
over(b,c).
over(c,d).

SOLUTION ■ is perfectly correct for theoretical logic programming.

↓ When written in PROLOG as

1) over(X,Z) :- over(X,Y), over(Y,Z).
2) over(a,b).
3) over(b,c).
4) over(c,d).

■ $\models_2$ over(a,b). or over(a,c).

TRY 2 SEE WHAT HAPPENS.

LECTURE 7
LOGIC PROGRAMMING

The program loops.

1 method to solve it.

```
over(X,Y) :- on(X,Y).
over(X,Z) :- on(X,Y), over(Y,Z).

on(a,b).
on(b,c).
on(c,d).
```

□

To see why the above happens we need to know

- what is SLD-RESOLUTION?
- how it is implemented in PROLOG?