LECTURE 3
LOGIC PROGRAMMING
PROPOSITIONAL LOGIC – CONTINUATION

SPECIAL CASES OF CLAUSES:

1. An empty head

(i) $\leftarrow P_1, P_2, \ldots, P_n$ CF

negative clause

(ii) $\underbrace{\sim P_1 \vee \sim P_2 \vee \ldots \vee \sim P_n}_{f}$ DNF

As $I(f \vee F) \equiv I(f)$

I ↑ interpretation

(i) is identical to:

(iii) $F \leftarrow P_1, P_2, \ldots, P_n$

$F \vee \sim P_1 \vee \sim P_2 \vee \ldots \vee \sim P_n$

2. An empty body

(i) $P_1, P_2, \ldots, P_n \leftarrow$ CF

(ii) $P_1 \vee P_2 \vee \ldots \vee P_n$ DNF

LECTURE 3
LOGIC PROGRAMMING

Again $I(P_1 \vee P_2 \vee .. \vee P_n) = I(P_1 \vee P_2 \vee .. \vee P_n \vee F)$, where $F \equiv \sim T$

(i) reads

(iii) $P_1, P_2, \ldots, P_n \leftarrow T$

3. An empty (contradiction) clause

(i) $\leftarrow \quad \equiv F \leftarrow T$

As $I(F \leftarrow T) = I(F \vee \sim T) = T(F \vee F) = false$

(i) is always false - contradiction

We denote it as $\blacksquare$.

.. <u>Positive</u> & <u>negative</u> literals

$P \leftarrow \quad \equiv P$

$\leftarrow P \quad \equiv \sim P$

LOGIC PROGRAMMING: deals only with <u>HORN CLAUSES</u>:

LECTURE 3

LOGIC PROGRAMMING

DEFINITION 1: **Horn clauses:** a set of clauses with at most one positive literals.

So we may have:

(i) $A \leftarrow B_1, B_2, \ldots, B_n$
(ii) $A \leftarrow$

] data clauses.

(iii) $\leftarrow B_1, B_2, \ldots, B_n$
(iv) $\leftarrow$

] clauses in execution of PROLOG program

Note: in the set of initial data i.e.

$$\Delta = \{clause\_1, \ldots, clause\_m\}$$

only clauses of type (Horn clauses)

(i) a rule
(ii) a fact

are permitted!

Horn clauses of type (iii) & (iv) will appear during the execution of PROLOG.

LECTURE 3
LOGIC PROGRAMMING

Example 1:

P, Q.	$\equiv P \vee Q$
:- P, Q.	$\equiv \sim P \vee \sim Q$
R, S :- P.	$\equiv R \vee S \vee \sim P$

Logic program in PROLOG $\equiv \Delta \leftarrow$ initial data

All clauses <u>are disallowed</u> in Δ^{NH}. PROLOG program
The third one is <u>not a Horn</u> clause.
The first one is <u>not a Horn</u> clause.
The second one is a Horn clause but not exactly with one atom in the head.
We can apply to Δ^{NH} all theory but not in PROLOG – explained later. □

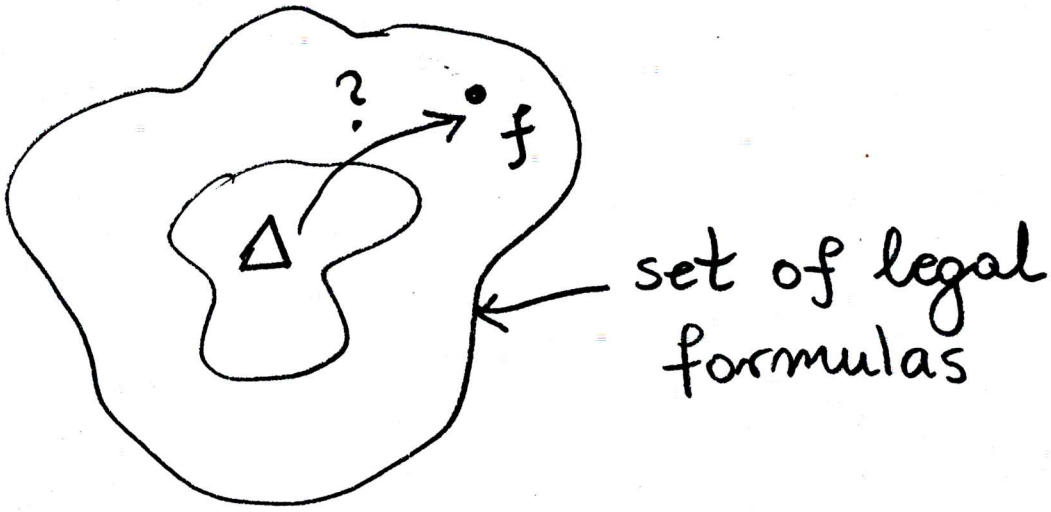
Remark: Be careful with the above.
Also each clause should be terminated with a dot.

LECTURE 3
LOGIC PROGRAMMING

We introduce now the notion of

logical consequence

which is for a given Δ & LOGIC independent from the available set of inference rules.



Intuition: does f "follow" from Δ?

First: a notion of a *model* is needed here.

LECTURE 3
LOGIC PROGRAMMING

DEFINITION 2: Let Δ be a finite set of legal Propositional Logic formulas. An **interpretation** I:

$$I : \Delta \longrightarrow \{true, false\}$$

which makes **all** $f \in \Delta$ **true** is called a **MODEL** (i.e. $\forall f \in \Delta \; I(f) = true$)

Example 2: Let $\Delta = \{p \wedge q, \; p \wedge r\}$

(i) $I(p) = I(q) = I(r) = true$ $\quad \Delta_a = \{p, q, r\}$

↖ ↑ ↑ this interpretation **is a model** for Δ as:

$$I(p \wedge q) = true.$$
$$I(p \wedge r) = true.$$

(ii) $I(p) = I(q) = true$ & $I(r) = false$

↖ ↑ ↗ this interpretation **is not a model** for Δ as $I(p \wedge r) = false$. □

LECTURE 3
LOGIC PROGRAMMING

DEFINITION 3: A legal formula Q is said to be **a logical consequence** of Δ:

$$\boxed{\Delta \models Q}$$ notation

if any model for Δ is a model for Q i.e:

$$\boxed{\forall f \in \Delta \quad I(f) = \text{true} \Rightarrow I(Q) = \text{true}.}$$

Remark: **PROLOG** will in fact reply which queried formulas are logical consequences.

For PROPOSITIONAL LOGIC we have a **simple** & **brute-force method** for determining the logical consequences:

LECTURE 3

LOGIC PROGRAMMING

Algorithm:

Step 0: create Δ_a from Δ ($\Delta_a \leftarrow$ set of propositional symbols from Δ)

Step 1: we list for Δ_a (derived from Δ) a list of all interpretations — always a finite set i.e. $2^{\overline{\overline{\Delta_a}}}$.

Step 2: we check which of those are models for Δ

Step 3: if any model for Δ is also a model for Q then we have:

$$\boxed{\Delta \models Q}.$$

If at least one model for Δ is not a model for $Q \Rightarrow \boxed{\Delta \not\models Q}$

Note that if there is **no model** for $\Delta \Rightarrow$ we regard this as $\Delta \models Q$.

LECTURE 3
LOGIC PROGRAMMING

Example 3:

$\Delta = \{ Q \leftarrow P \; ; \; P \}$

Modus Ponens yields Q

$\Delta_a = \{ Q, P \}$

$2^{\overline{\Delta_a}} = 2^2 = 4$ ← interpretations

(*) $\Delta \overset{?}{\models} Q$

	P	Q	Q ← P	P	Q
I_1	0	0	1	0	0
I_2	0	1	1	0	1
I_3	1	0	0	1	0
I_4	1	1	1	1	1

for Δ_a | for Δ | for query (*)

I_4 is the only model for Δ.

But also it is a model for Q!

⇓

$\Delta \models Q$

SOUNDNESS of MP.

MODUS PONENS is a valid inference as it produces logical consequences. □

LECTURE 3

LOGIC PROGRAMMING

Example 4: The following inference I_F is **not valid** (not sound):

$$I_F \Downarrow \frac{P, Q \leftarrow R \quad ; \quad Q}{P \leftarrow R}$$

$\Delta = \{ P, Q \leftarrow R \; ; \; Q \}$

$\Delta_a = \{ P, Q, R \}$

The number of all interpretations $2^{\overline{\overline{\Delta_a}}} = 2^3 = 8$

	P	Q	R	$P, Q \leftarrow R$	Q	$P \leftarrow R$
I_1	0	0	0	1	0	1
I_2	0	0	1	0	0	0
I_3	0	1	0	1	1	1
I_4	0	1	1	1	1	0
I_5	1	0	0	1	0	1
I_6	1	0	1	1	0	1
I_7	1	1	0	1	1	1
I_8	1	1	1	1	1	1

for Δ_a | for Δ | for queried $P \leftarrow R$

I_3, I_4, I_7, I_8 are models for Δ.

But I_4 is not a model for $P \leftarrow R$.

Hence $\Delta \not\models P \leftarrow R$

LECTURE 3
LOGIC PROGRAMMING

"Truth table" method based on definition of logical consequence:

- is effective
- but not sophisticated (exponential growth of rows in truth table!)

Thus using truth-table to verify

logical consequence

soon becomes

impractical!

One should consider other techniques to prove logical consequences.

An alternative: introduce some sound inference rule (producing logical consequence) – a valid inference.

LECTURE 3

LOGIC PROGRAMMING

Important from the computer point of view as:

- finding logical consequences does not require any particular knowledge about specific situation, but merely the mechanical application of the rules of inference
- in particular we want to find logical consequences of Δ built from clauses.

DEFINITION 4: Let $\{A_i\}_{i=1}^{m}$ & $\{B_j\}_{j=1}^{n}$ be literals

$$f_1 \equiv L \vee A_1 \vee A_2 \vee \dots \vee A_m$$

and

$$f_2 \equiv \sim L \vee B_1 \vee B_2 \vee \dots \vee B_n$$

then **RESOLUTION** is the inference yielding a new clause $R(f_1, f_2) = f$:

$$\boxed{f \equiv A_1 \vee \dots \vee A_m \vee B_1 \vee \dots \vee B_n} \quad (\blacktriangle)$$

LECTURE 3
LOGIC PROGRAMMING

Note:

- we will generalize the notion of a clause for a PREDICATE LOGIC
- accordingly the notion of RESOLUTION acting on clauses will be later extended to PREDICATE LOGIC

RESOLUTION can be expressed in terms of "clausal notation" i.e.:

$$P_1, P_2, \ldots, P_k, L \leftarrow S_1, S_2, \ldots, S_t \qquad (*)$$

$$Q_1, Q_2, \ldots, Q_m \leftarrow L, R_1, R_2, \ldots, R_n \qquad (**)$$

$$(***)\ P_1, P_2, \ldots, P_k, Q_1, Q_2, \ldots, Q_m \leftarrow S_1, S_2, \ldots S_t, R_1, R_2, \ldots, R_n$$

Note that position of L in (*) & (**) is irrelevant as semantics of disjunction is commutative i.e. $I(p \vee q) = I(q \vee p)$.

interpretation

The resulting clause (***) (or (***)) is called

RESOLVENT

LECTURE 3
LOGIC PROGRAMMING

Note: in particular **RESOLUTION** covers **Modus Ponens** (MP) & **Modus Tolens** (MT)

MP
$$\frac{Q \leftarrow P \quad P}{Q} \equiv \frac{Q \vee \sim P \quad P}{Q} \text{ RESOLUTION}$$ with $L = P$

MT
$$\frac{Q \leftarrow P \quad \sim Q}{\sim P} \equiv \frac{Q \vee \sim P \quad \sim Q}{\sim P} \text{ RESOLUTION}$$ with $L = Q$

Of course **RESOLUTION** does not cover **AND ELIMINATION** (since $p \wedge q$ is not a clause & **RESOLUTION** cannot be applied).

We justify now the

"validity" of RESOLUTION

for the clauses from PROPOSITIONAL LOGIC.

LECTURE 3
LOGIC PROGRAMMING

Two <u>important questions</u> arise with any inference rule:

a) <u>SOUNDNESS</u> of the inference rule:

Does the inference rule produce always logical consequences?

b) <u>COMPLETENESS</u> of the inference rule:

Can a given inference rule generate all possible logical consequences of given data?

More formally for $\mathcal{J} = \{I_1, I_2, \ldots, I_n\}$ being a set of inference rules & Δ a set of legal formulas

a) soundness of a given $I_k \in \mathcal{J}$:

$$\text{if } \Delta \overset{I_k}{\vdash\!\!\rightarrow} Q \text{ then } \Delta \models Q \quad ?$$

where $\Delta \overset{I_k}{\vdash\!\!\rightarrow} Q$ denotes conclusion Q derivable from Δ via I_k inference rule.

b) completeness of a given $I_k \in \mathcal{J}$:

$$\text{if } \Delta \models Q \text{ then } \Delta \overset{I_k}{\vdash\!\!\rightarrow} Q \quad ?$$

LECTURE 3
LOGIC PROGRAMMING

When we say **validity** of a given inference rule we mean **SOUNDNESS**.

PROLOG:

$\Delta \equiv$ set of Horn clauses* with exactly one positive literal (non-empty head)

$\mathcal{I} = \{$ RESOLUTION $\}$.

Example 5: We may have inferences which <u>do not satisfy</u> a) or b)

a) <u>NOT SOUND</u>: introduce an inference

I_{NS}: **If p∨q then p**

clearly for $\Delta = \{p \vee q\}$ $I(p) = \text{false}$, $I(q) = \text{true}$ is a model for Δ.

But I is not a model for q.

Thus $\Delta \not\models q = I_{NS}(p \vee q)$. Hence I_{NS} is <u>not sound</u>.

b) <u>NOT COMPLETE</u>:

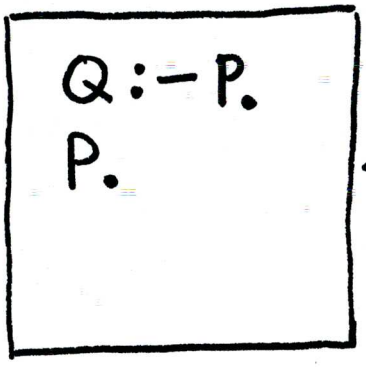
Let $\Delta = \{p \wedge q\}$ & $\mathcal{I} = \{$RESOLUTION$\}$

* generalized to PREDICATE LOGIC Horn clauses

LECTURE 2
LOGIC PROGRAMMING

Step 1: we write a program in PROLOG

example.pl ← any name.pl



←→this represents data Δ

Step 2: we open PROLOG session by invoking PROLOG interpreter.

a) > pl ⊕RTN Linux shell session

b) >> ← prolog session opens.

c)>>consult('example.pl'). ⊕RTN
← invoking data

d)>>Q. ⊕RTN
← asking PROLOG whether Q can be inferred.

LECTURE 3
LOGIC PROGRAMMING

Clearly $\Delta \models p$ as any model for Δ is a model for p.

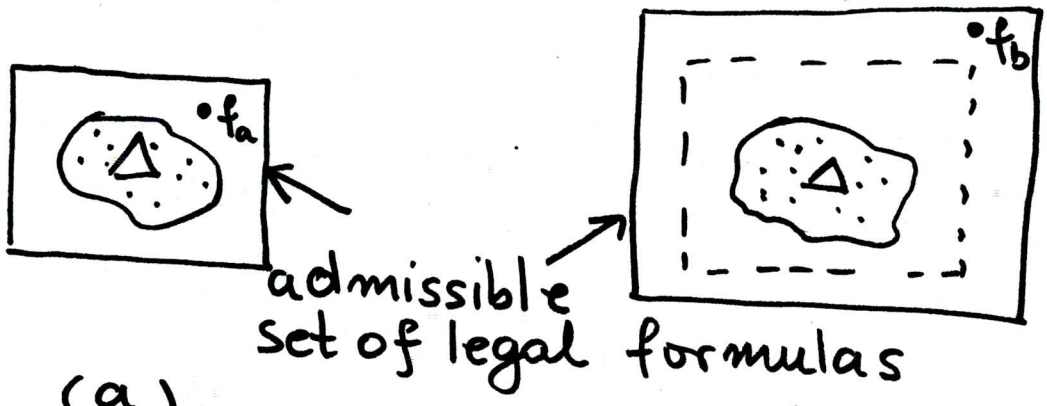
But we cannot infer p from Δ via RESOLUTION i.e. $\Delta \not\vdash_{\text{RESOLUTION}} p$

as Δ does not contain clauses and RESOLUTION does not operate on conjunction

□

Note:

Completeness depends on initial set of data and on the set of legal formulas we search for logical consequences



(a) (b)

In (a) we may have completeness but in (b) > (a) we may lose it!

LECTURE 3
LOGIC PROGRAMMING

as $\forall f_a \in (a) \quad \Delta \models f_a$ then $\Delta \underset{I}{\vdash\!\!\sim\!\!>} f_a$

but we may have $f_b \in (a) \setminus (b)$ (set difference) such that

$$\Delta \models f_b \quad \text{but} \quad \Delta \not\underset{I}{\vdash\!\!\sim\!\!>} f_b.$$

Note: for arbitrary clause from $\Delta \ni f_c$ from definition we have

$$\Delta \models f_c$$

and also as $\quad f_c \vee T \equiv T$

$$f_c \vee \sim T \equiv f_c$$

$$\text{RESOLUTION}(f_c, T) = f_c.$$

Thus $\quad \Delta \underset{\text{RESOLUTION}}{\sim\!\!>} f_c.$

So whatever is in Δ it is a logical consequence of Δ (also it is our intuition) & if Δ is set of clauses each clause is derivable from Δ !

LECTURE 3
LOGIC PROGRAMMING

We close this lecture by showing the validity (soundness) of RESOLUTION*.

Theorem 1.

The following inference (A RESOLUTION)

$$\frac{\begin{array}{l} L \vee A_1 \vee A_2 \vee \ldots \vee A_m \equiv f_1 \\ \sim L \vee B_1 \vee B_2 \vee \ldots \vee B_n \equiv f_2 \end{array}}{A_1 \vee A_2 \vee \ldots \vee A_m \vee B_1 \vee B_2 \vee \ldots \vee B_n \equiv f_3,} \text{ RESOLUTION}$$

where A_i $(1 \leq i \leq m)$ & B_j $(1 \leq j \leq n)$ are literals, is <u>a valid inference</u> (i.e. SOUND)

PROOF:

Suppose we have at least one model for Δ – i.e. I_M.

As $f_1, f_2 \in \Delta$ & as I_M is a model for Δ then

$$I_M(f_1) = I_M(f_2) = \text{true}$$

Case 1: $I_M(L) = \text{true}$.

Thus $I_M(\sim L)$ false

Hence as $I_M(f_2) = \text{true} \Rightarrow I_M(B_1 \vee \ldots \vee B_n) = \text{true}$

* For now only with the framework of PROPOSITIONAL LOGIC.

Therefore

$$I_M(f_3) = \text{true}.$$

So I_M is a model for the RESOLVENT f_3.

Case 2: $I_M(L) = \text{false} \Rightarrow I(A_1 \vee \ldots \vee A_m) = \text{true}$ & the proof follows the previous pattern.

So for each model of Δ we showed $I_M(f_3) = \text{true}$.

So $\boxed{\Delta \models f_3.}$

If there is no model for Δ then by definition

$$\boxed{\Delta \models f_3.}$$

The proof is complete.

RESOLUTION IS SOUND (VALID)

COMPLETENESS will be shown later for more general case (Predicate Logic).

PROLOG FINDS LOGICAL CONSEQUENCES OF THE DATA FORMATTED IN TERMS OF CLAUSES.