LECTURE 1

LOGIC PROGRAMMING

INTRODUCTION

a) & b) allow to produce

⇓

"new conclusions"

Fundamental questions:

(i) are "new conclusions"* generated from a) & b) "logical consequences"** of the initial data?

Can we produce SOUND conclusions?

* the notion of "new conclusion" depends on data & inference rule

** the notion of "logical consequence" is independent from inference rule

LECTURE 1
LOGIC PROGRAMMING

set of "new conclusions" from $\{\Delta, \mathcal{J}\}$

Δ = Data $\mathcal{J}$ = set of inference rules

$\Delta_{NC}^{\mathcal{J}}$ = set of "new conclusions"

Δ_{LC} = set of "logical consequences"

QUESTION (i) $\boxed{\Delta_{NC}^{\mathcal{J}} \subseteq \Delta_{LC}}$

(ii) are all "logical consequences" obtainable from "new conclusions"

QUESTION (ii) $\boxed{\Delta_{LC} \subseteq \Delta_{NC}^{\mathcal{J}}}$

LECTURE 1
LOGIC PROGRAMMING

QUESTION (i):

important as we wish to draw **sound** conclusions.

QUESTION (ii):

important to maximize the power of $\{\Delta, \mathcal{J}\}$ to reach all logical consequences **completeness**.

In order to build a system on computer we need to:

a) formally define formatted data $\Delta = \{f_1, f_2, \ldots, f_n\}$

b) impose some set of **inference rules**

$$\mathcal{J} = \{I_1, I_2, \ldots, I_m\}$$

operating on Δ & producing "logical consequences"

c) get as much as possible of all logical consequences with $\{A, J\}$

d) build a system in computer which is computationally feasible

e) provide a friendly programming language

f) find in one query more than one "logical consequence"

We have two fighting demons:

LECTURE 1
LOGIC PROGRAMMING

"GOOD" wants:

- as much as possible big set of admissible data & inference rules (to assure (ii))

"BAD" wants:

- as little as possible admissible set of data & inference rules (to maintain computational feasibility of the implemented system)

QUESTION:

where is the trade-off ?

LOGIC PROGRAMMIG addresses this issue !

LECTURE 1
LOGIC PROGRAMMING

Let us look at possible set of some inference rules:

Example 1.

Rule: if it is raining then I will get wet.

Fact: it is raining.

If:

a) heuristic standing behind Rule

b) Fact eventuates

 then

it is reasonable to conclude:

New Fact: I will get wet.

□

EXAMPLE 2.

RULE: if inflation is increasing then the price of petrol will rise.

Fact: inflation is increasing.

Now if we accept RULE as sound & the Fact to take place

⇓

one concludes:

New Fact: the price of petrol will rise.

□.

We can sum up Ex 1 & Ex 2.

Table 1.

Rule:	if P is true then Q is true
Fact:	P is true
	⇓
New Rule:	Q is true

This inference rule is called:

MODUS PONENS ($MP \equiv I_1$)

Another example of inference rule:

Table 2.

Rule: if P is true then Q is true.
Fact: Q is false.
⇓
New Rule*: P is false.

This inference rule is called:

MODUS TOLENS ($MT \equiv I_2$)

↑

it complements Modus Ponens

* otherwise by MP we would have Q to be true – contradition.

LECTURE 1
LOGIC PROGRAMMING

Another example of inference rule:

Fact: P & Q is true

$\Downarrow$

New Fact 1: P is true

New Fact 2: Q is true

This inference rule is called

AND ELIMINATION ($AE \equiv I_3$)

There are many other inference rules which are "sound" e.g.

UNIVERSAL INSTANTATION ($UI \equiv I_4$

Fact: $Q(x)$ is true for each x in a given domain D
$\forall x Q(x)$

$\Downarrow$

New Fact: for each $x_0 \in D \Rightarrow Q(x_0)$ is true.

LECTURE 1

LOGIC PROGRAMMING

EXISTENTIAL INSTANTATION (EI=I)

Fact: there exists x that $Q(x)$ is true $(\exists x\, Q(x))$ over domain D

$\Downarrow$

New Fact: we can infer $Q(x_0)$ for some x_0 from a domain D

Example 3. a) $\forall x \in \mathbb{R}$ $\underbrace{x^2 \geq 0}_{Q(x)}$ true

$\Rightarrow$ whatever $x_0 \in \mathbb{R}$ $x_0^2 \geq 0$. (UI)

b) $\exists x \in \mathbb{R}$ $x > 0$ true $\Rightarrow$ there exists at least one $x_0 \in \mathbb{R}$ $x_0 > 0$ (EI)

□

Any inference rule is good provided

- it conforms to common sense
- it works on properly defined set of data

LECTURE 1
LOGIC PROGRAMMING

LOGIC: is a branch of mathematics that deals with expressing statements about real world in

- a suitably precise symbolic form
- and in processing those symbols using suitably chosen inference rules.

Hope & TASK WITH COMPUTER:

- to provide statements for computer (data)
- to equip it with inference rule(s)

⇓ we expect

computer will deduce other statements with <u>no understanding</u> of what they mean.

LECTURE 1
LOGIC PROGRAMMING

AIMS OF LOGIC PROGRAMMING :

- give the computer a data base of facts & rules about a given situation

- query the computer as to which new facts it can deduce

A computer solution to a problem consists of two components:

- the **logic of the problem** (declarative aspect – it describes exactly what the problem is)

- the **control of the problem** (procedural aspect – it describes how the problem is to be solved)

LECTURE 1
LOGIC PROGRAMMING

In programming in procedural language (like Pascal, C, Fortran etc) the programmer must specify all steps to be taken by a computer.

LOGIC PROGRAMMING is purely

a declarative programming

i.e. the programmer merely specifies the problem & leaves the control to the computer.

LOGIC PROGRAMMING is used in:

- expert systems
- artificial intelligence
- automatic theorem proving
- machine learning

LECTURE 1

LOGIC PROGRAMMING

DEFINITION 1:

Given a set of inference rules:

$$\mathcal{J} = \{I_1, I_2, \ldots, I_n\}$$

we say that a conclusion ϕ is derivable from a set of premises

$$\Delta = \{f_1, f_2, \ldots, f_m\}$$

via $\mathcal{J}$ if and only if:

either

a) $\phi \in \Delta$ (is a member of Δ)

or

b)* ϕ is a result of applying of any inference rule $I_i \in \mathcal{J}$ to sentences derivable from Δ (in a finite number of steps)

Notation for a), b)

$$\Delta \overset{\mathcal{J}}{\rightsquigarrow} \phi$$

(*) inductive component of Definition 1.

LECTURE 1
LOGIC PROGRAMMING

In particular if only <u>one inference rule is used</u> i.e. $I_k \in \mathcal{J}$ only, we say that ϕ is **derivable** from Δ via I_k:

$$\Delta \overset{I_k}{\rightarrowtail} \phi$$

via single inference rule

Example 4. Let $\Delta = \{\phi \wedge \psi, \psi \rightarrow \Omega\}$ & $\mathcal{J}_a = \{MP, AE\}$

Question: $\Delta \overset{i}{\underset{\mathcal{J}_a}{\rightarrowtail}} \Omega$

<u>1. Step:</u> $\phi \wedge \psi$ & $AE \Rightarrow$ ϕ true & ψ true

(so $\Delta \overset{}{\underset{\mathcal{J}_a}{\rightarrowtail}} \phi$ & $\Delta \overset{}{\underset{\mathcal{J}_a}{\rightarrowtail}} \psi$)

<u>2. Step:</u> $\psi \rightarrow \Omega$, ψ & $MP \Rightarrow \Omega$ true

Thus $\Delta \overset{}{\underset{\mathcal{J}_a}{\rightarrowtail}} \Omega$.

LECTURE 1

LOGIC PROGRAMMING

Note that both AE & MP are needed.

If we change $\mathcal{J}_a$ to $\mathcal{J}_b = \{MP\}$

$\Rightarrow$ $\Delta \not\vdash_{\mathcal{J}_b} \Omega$

is not derivable from Δ via $\mathcal{J}_b$

$\mathcal{J}_b$ is here too thin! □

Here we allowed a BAD demon to have too much influence on cutting the capacity of inferring new formulas ($\mathcal{J}_b$ is <u>too thin</u>).

In LOGIC PROGRAHMING we choose:

(*)

a) $\Delta \leftarrow$ set of the so-called **<u>clauses</u>**

b) $\mathbf{I} = \{$ one inference rule – **<u>RESOLUTION</u>** $\}$

(*) both formally defined later

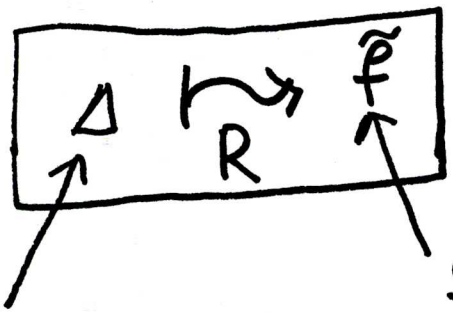
LECTURE 1
LOGIC PROGRAMMING

RESOLUTION (clause1, clause2) = clause
↑ R ↑ new

binary inference rule.

For **PROLOG** (Programming Logic Language)

we show (roughly speaking):



set of clauses

specially formatted formulas*

(*) explained later what is the format of $\tilde{f}$. Due to the indirect proof it is more than clause