Logic and Inference
Rules

Introduction

Knowledge representation

- Has been studied long before the emerge of the WWW in the field of artificial intelligence, or philosophy(Aristotle, the father of logic)
- Logic is the foundation of knowledge representation

Why is logic important for us?

- It provides high-level language
- It has well understood formal semantics
- There exists precise notion of logical consequence
- There exists proof systems which make possible to trace the proof that leads to a logical consequence
- It provides explanations for answers

Predicate logic

Can be divided in subsets:

Description logic

- Corresponds to OWL Lite and OWL DL

Rules systems (Horn logic, definite logic programs)

Rules

A rule has the form: A1,A2,…,An→B

where A1,…,An,B atomic formulas

There are two interpretations of this rule:

- If A1,…,An are known to be true, then B is also true
- If the conditions A1,…,An are true, then carry out the action B

Observation1

Description logics and Horn logic are orthogonal

Example

A person studies and lives in the same city.

We want:

- to say that this person is “home student”
- to say that the person is either man or woman

Rules Clasification(1)

There are different types of rules

Example

A vendor who wants to give special discount if it is customer ´s birthday

R1:If birthday, then special discount

R2:If not birthday, no special discount

The system works when the provided information is complete

Rules Clasification (2)

What if customer ´s birthday is unknown?

R1:If birthday, then special discount

R2:If birthday unknown, no special discount

This system of rules is applied when information is incomplete

Rules Clasification(3)

- MONOTONIC Rules

-remain valid even when new knowledge becomes available

- NONMONOTONIC Rules

-becomes invalid when new information is added

Monotonic Rules(1)

Example: Family relations

Base predicates:

Mother(X,Y)-X is mother for Y

Male(X)-X is male

Derived predicates:

Male(X),Parent(P,X),Parent(P,Y),Notsame(X,Y)
→Brother(X,Y)

Monotonic Rules(2)

Syntax:

loyalCustomer(X),age(X)>60→discount(X)

Variables: X

Constants: 60

Predicates: loyalCustomer

Function symbols : age

Monotonic Rules(3)

Rule: A1,A2,…,An→B

We call:

B-head of the rule

{A1,…,An} –body of the rule

A1,A2,…-devices

Monotonic Rules(4)

Facts

- LoyalCustomer(b23458) says that the customer with the given id is loyal.

Logic Programs

A program is a finite set of facts and rules

Goals

A goal denotes a query asked to a logic program

Nonmonotonic Rules(1)

- In the nonmonotonic rule system, one rule may not be applied, even if all the premises are proven

Example

p(X)→q(X)

r(X)→non(q(X))

CONFLICT!!!!!

Nonmonotonic Rules(2)

The conflict can be resolved using priorities among rules

We can use the following notation to specify the fact that the rule r1 is stronger than r2.

r1: p(X)→q(X)

r2: r(X)→non(q(X))

r1> r2

Nonmonotonic Rules(3)

Note:

It is required that the priority relation to be acyclic

Syntax

r : L1,L2,…,Ln →L

L –the head of the rule

{L1,…,Ln}-the body of the rule

L1,L2,…-literals

Nonmonotonic Rules(4)

- *Defeasible Logic Program*

Is a triple (F,R,>)consisting of:

a set of facts F

a finite set of defeasible rules

an acyclic binary relation

Nonmonotonic Rules(5)

An example

- Imagine that we want to rent an apartment, which should:
  - be at least 45sq m
  - have at least 2 bedrooms
  - if it on the 3rd floor or higher, there must be a lift
  - pets allowed

rent offered :300 euro in the city centre ,
250 euro suburbs

You are willing to pay 5 euro/sq m for a larger appartment and 2 euro/sq m of garden

The total amount shouldn ´ t be bigger than 400

Priorities:
-the price
-the presence of garden
-additional space

Formalisation of requirments

- size(x,y) – y is the size of apartment x
- bedrooms(x,y) – x has y bedrooms
- price (x,y) - …
- floor(x,y) - …
- garden(x,y) - …
- lift(x) - …

(2)

- pets(x) - …
- central(x) - …
- acceptable(x) – flat x satisfies our requirements
- offer(x,y) – we pay the amount y for the flat x

(3)

Rules:

- r1:➔ acceptable(x)
- r2:bedrooms(x,y),y<2 ➔not acceptable(x)
- r3:size(x,y),y<45➔not acceptable(x)
- r4:not pets(x) ➔not acceptable(x)
- r5:floor(x,y),y>2,not lift(x) ➔not acceptable(x)
- r6:price(x,y),y>400➔not acceptable(x)

(4)

Calculating the offer for an apartment

r7:size(x,y),y>45,garden(x,z),central(x)

➔offer(x,300+2z+5(y-45))

r8:size(x,y),y>45,garden(x,z),not(central(x))

➔offer(x,250+2z+5(y-45))

r9:offer(x,y),price(x,z),y<z➔not acceptable(x)

(5)

After finding the acceptable apartments,we need to select one according to our preferences

- r10:cheapest(x)➔rent(x)
- r11:cheapest(x),largestgarden(x)➔rent(x)
- r12:cheapest(x),largestgarden(x),largest(x)

➔rent(x)

r12>r11>r10

Rule Markup in XML
-*Monotonic Rules(1)*

- Terms

Used tags :<term>,<function>,<var>,<const>

f(x,a,g(b,y))

- Atomic formulas

Additonal tags:<atom>,<predicate>

p(x,a,f(b,y))

-*Monotonic Rules(2)*

- Facts

An atomic formula enclosed by <fact> tags

- Rules

Head: an atomic formula

Body: a sequence of atomic formulas

p(x,a),q(y,b)→r(x,y)

-Monotonic Rules(3)

- Queries

Are represented as bodies of rules surrounded by

<query> tags

- A DTD

<!ELEMENT program((rule|fact)*)>

<!ELEMENT fact(atom)>

<!ELEMENT rule (head,body)>

<!ELEMENT head(atom)>

-Monotonic Rules(4)

<!ELEMENT body(atom*)>

<!ELEMENT atom(predicate,term*)>

<!ELEMENT term(const|var|function,term*))>

Predicates,function symbols,constants,variables are atomic types

<!ELEMENT predicate (#PCDATA)

<!ELEMENT function (#PCDATA)

<!ELEMENT var (#PCDATA)

Rule Markup in XML
-Nonmonotonic Rules

Diffrences between non- and monotonic rules:

Nonmonotonic rules have:

- No function symbols
- Negated atoms may occur in the head and the body of the rule
- Each rule has a label
- A programs contains priority statements

Examle of defesible program

r1: p(x)➔s(x)

r2: q(x)➔non s(x)

p(a) ;q(a)

r1 > r2

Representation: <rule id = „r1“>…

-priorities : <stronger superior=„r1“ inferior=„r2“>

…

A DTD(1)

<! ELEMENT program ((rule|fact|stronger)*)>

<! ELEMENT fact (atom|neg)>
<! ELEMENT neg (atom)>

<! ELEMENT rule|(head,body)>
<ATTLIST rule
id ID #IMPLIED>

A DTD (2)

<! ELEMENT head (atom|neg)>
<! ELEMENT body ((atom|neg)*)>

Priorities:
<! ELEMENT stronger (EMPTY)>
<!ATTLIST stronger
superior IDREF #REQUIRED
inferior IDREF #REQUIRED

A DTD (3)

Predicates,constants and variables are atomic types

<! ELEMENT predicate (#PCDATA)>

<! ELEMENT var (#PCDATA)>

<! ELEMENT const (#PCDATA)>