Lecture 4

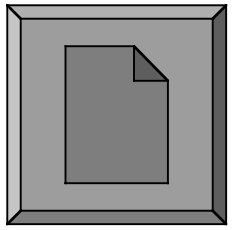
308-424A

Topics in AI

Gregory Dudek

Predicate calculus cont'd

- Review & new stuff.



We discussed soundness, completeness, semi-decidability.
We looked at examples illustrating the syntax of FOL and Generalized Modus Ponens.

Clausal Form

- Any predicate calculus proposition can be converted into clauses in 6 steps:
  - Removing implications
  - Moving negation inwards
  - Skolemising (remove quantifiers)
  - Moving universal quantifiers outwards
  - Distributing AND over OR (for CNF)
  - Putting into clauses (notation only)
- **Read R&N Sec 9.6 (p. 281) or Dean,Allen,Aloimonos Sec 3.5 pp. 96**

The Horn (Clause) of Plenty

- We have observed an equivalence between arbitrary sentences in FOL and CNF.
  - This extends to **HORN CLAUSES**
    - R&N discusses Horn Sentences in Sec. 6.4
- Thus, we can use Horn clauses to expresses anything in FOL.
- The PROLOG language uses Horn clauses explicitly as its notation.

PROLOG

- PROLOG: a logic programming language.
  - Name derives from **PROgramming in LOGic**
- Based on theorem proving, first-order logic.
- Small, unusual, influential language.
- Developed in the 1970s.
- On-line: documentation and executables for SOCS (lisa) and home PCs.

PROLOG notation

AND	, (comma)
OR	; (semicolon)
IMPLIES	:-
NOT	not

Variables start with uppercase

Predicates & bound variables in lower case

Facts & rules

- In prolog we can state facts like natasha likes nicholas by defining a suitable predicate:
  - Likes(natasha, nicholas)
- We can define rules that allow inference.
- Uses **closed world assumption**: *anything is false unless it is provably true.*

This is an important idea, even outside of the prolog context.

Rules

- Rules:
  - One predicate as conclusion.
    - Implication works *to the left*.
    - Left hand predicate must be a positive literal.
  - Resolution and unification are the “internal” mechanisms.
- Prolog is based on satisfying goals using a resolution theorem prover.

PROLOG Examples

```
likes(X,dudek)
likes(Everybody,cs424)
```

Everybody likes cs424

```
Likes(richard,X), likes(eric,X)
```

Things likes by both richard and eric

```
Likes(phil,X) :- likes(eric,X)
```

If eric likes something, so does phil.

The Montreal Student Domain

```
goodstudent(X) :- awakeinclass(X), csstudent(X).
csstudent(X) :- smart(X), (adventurous(X) ;
  sensible(X)).
adventurous(X) :- ( montrealer(X) ;
  rockclimber(X) ).
awakeinclass(X) :- drinks(X,Y) , hasdrug(Y,Z),
  stimulant(Z).
smart(X) :- not(rockclimber(X)), reader(X).
```

Facts about people.

- `montrealer(jane).`
- `smart(jane).`
- `nerd(jane).`
- `drinks(jane,coffee).`

- `montrealer(bob).`
- `nerd(bob).`
- `drinks(bob,sprite).`

- `owns(teapot,ted).`

More facts...

- `reader(ted).`
- `reader(mary).`
- `reader(helen).`

- `fatherof(mary,ted).`

- `drinks(helen,sprite).`

```
hasdrug(tea,caffiene).
hasdrug(tea,tannin).
hasdrug(tea,theobromine).
hasdrug(coffee,caffiene).
hasdrug(coffee,oil).
hasdrug(quat,foo).
hasdrug(sprite,sugar).
stimulant(caffiene).
% stimulant(theobromine).
```

Simple stuff

- `reader(ted).`

```
yes
```

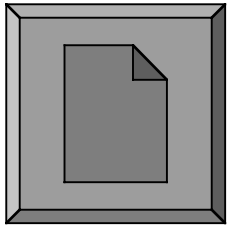
- `reader(X).`

```
X = ted ;

X = mary ;

X = helen ;

no
```

On-line examples...

WE NEVER MANAGED TO DO THIS DUE TO THE EQUIPMENT PROBLEMS.

Run prolog



We will consult a prolog program file called “mtl” and then make some queries.

This is “open prolog” for the Mac.