1

Knowledge Based Systems

(CM0377)
Lecture 4

(Last modified 5th February 2001)

2

So Prolog looks cool, but ...

- It's an implementation of clausal logic, so:
  - what are the limits of expressiveness of clausal logic?
  - what are the limits of reasoning with clausal logic? (what can/cannot be (effectively) computed?)
  - How are these two limitations related? Can we enhance, e.g., reasoning/efficiency by limiting expressiveness?
- How do the more esoteric kinds of reasoning (e.g. induction) relate to the traditional reasoning we have seen illustrated so far?

For all this we need some theory!

3

3 aspects of the theory

- **Syntax** - language we're using: 'alphabet', 'words' and 'sentences'
- **Semantics** - meaning of words & sentences
- **Proof theory** - how to obtain new sentences (theorems) from assumed ones (axioms) by symbol manipulation (inference rules)

4

Main questions

- Is our inference rule *sound?* (truth of theorems assured by truth of axioms)
- Is it *complete?* (powerful enough to prove any possible theorem, given enough time)
- Our starting point will be propositional clausal logic & gradually build up to the (more complex) clausal logic of Prolog

5

Propositional clausal logic - syntax

- Propositions are denoted by *atoms* (single words starting with lowercase letter, possibly including an underscore)
- e.g.
  - `boy`
  - `girl`
  - `child`

6

Syntax (ctd.)

- ':-' - 'if'
- ';' - 'or'
- ',' - 'and'
- Clause comprises a *head,* followed by the ':-' symbol, followed by the *body*
- *Head* is a *disjunction* of atoms (with ';' in between); *Tail* is a *conjunction* of atoms (with ',' in between) e.g.

```
has_football; impoverished :- boy, good_footballer
```

7

Syntax - *programs*

- A *program* is a set of clauses, each terminated by a period ('.'). e.g.

```
girl; boy:-child.
child:-girl.
child:-boy.
```

8

Semantics - *Herbrand bases/interpretations*

- *Herbrand base* of a program *P* is the set of atoms occurring in *P*. (In program on previous slide, it's {`girl, boy, child`})
- *Herbrand interpretation* is a mapping from the *Herbrand base* into the set of truth values {**true**, **false**}, e.g. {`girl`→**true**, `boy`→**false**, `child`→**true**} or (just listing atoms assigned to **true**) {`girl, child`}.

9

Semantics - truth values of clauses

- A Herbrand interpretation assigns truth values to every atom; therefore to the clause as a whole.
  - Body of a clause is **true** if *every* atom in it is **true**
  - Head of a clause is **true** if *at least one* atom in it is **true**
- Clause is assigned truth value **true** if body of clause is **false** or head of clause is **true** (or both)
- (NB this means that **false:-false** and **true:-false** are both assigned the value **true**. The idea is that if the body is **false** then it's valid either for the head to be **true** or **false**, without it contradicting the ':-')

10

Semantics - models

- If a clause is **true** in an interpretation then the interpretation is a *model* for that clause.
- An interpretation is a model for a program if it is a model for each clause it contains.

11

Semantics - models

- E.g. consider the program:

```
girl; boy:-child.
child:-girl.
child:-boy.
```

- Models are ∅ (empty model), {`girl`, `child`}, {`boy`, `child`} and {`girl`, `boy`, `child`}.
- So this means we have ruled out the possibility that, e.g., the person is a girl and a boy but not a child, or that the person is a girl but not a boy or a child.

12

Logical consequence

- Now consider the program:

```
girl; boy:-child.
child:-girl.
child:-boy.
boy.
```

- Models are {`boy`, `child`} and {`girl`, `boy`, `child`}. As these are all models of the clause:

```
child.
```

- It follows that this new clause is a *logical consequence* of the program.
- In general, a clause C is a logical consequence of a program P if every model of the program is also a model of the clause. Signified $P \models C$.

13

Minimal model semantics

- The program doesn't have enough information to distinguish between {`boy`, `child`} and {`girl`, `boy`, `child`}. Clearly here we don't want to say someone's both a girl and a boy.
- In general, the principle is that we don't want to assume that anything is true which doesn't have to be (e.g. train timetable: don't have to list times trains *aren't* going!)

14

Minimal model semantics

- Any model for which no subset of it is a model is a *minimal model*
- In the above example we have just one. Here, we have *two* minimal models:

```
girl; boy:-child.
child.
```

- (namely {`boy`, `child`} and {`girl`, `child`}).
- If program contains only definite clauses (with single literal in head), then it can be proved that we can guarantee a unique minimal model.

15

Proof theory

- Checking that $P \models C$ by determining every model of P and checking it's a model of C is going to take a long time!
- So we want *inference rules* that can *derive* C from P. These inference rules are *syntactic,* dealing with the symbols only, so we need to prove that C is derived from P if, and only if, $P \models C$.

16

Resolution

- Suppose we have two clauses:
  $A; A_{1,1}; \ldots; A_{1,i}$:- $B_{1,1}, \ldots, B_{1,j}$
  $A_{2,1}; \ldots; A_{2,k}$:- $B_{2,1}, \ldots, B_{2,m}, A$
- We *select* a literal to resolve on, which appears in the head of one clause and body of the other - in this case, w.l.o.g., *A*
- The resolvent is
  $A_{1,1}; \ldots; A_{1,i}; A_{2,1}; \ldots; A_{2,k}$:-
  $B_{1,1}, \ldots, B_{1,j}, B_{2,1}, \ldots, B_{2,m}$
- except that if a literal occurs in the head of *both* clauses, it occurs only once in the resolvent; similarly if it occurs in the body

17

Resolution - example

- So resolving the following clauses on has_football:

```
happy:-boy, has_football.
has_football; impoverished:-boy,
  good_footballer.
```

- gives resolvent:

```
happy; impoverished:-boy,
  good_footballer.
```

- Such a resolvent can be used as input for another resolution step. A *proof (derivation)* of clause *C* from *P* is a sequence of such steps resulting in *C*. Write: $P \vdash C$.

18

Next time ...

- We'll ask whether what can be proved corresponds with what are the logical consequences of a program
- And we'll discuss the idea of using resolution for *refutation*
- Then we'll talk about more complicated kinds of clausal logics