Logic Programming Alan Smaill
29/11/2015

Tutorial for week 7 (November 2–6)

1. The MP (modus ponens) inference rule is *sound*, i.e. if the formulae above the line (assumptions) are true, then so is the conclusion (below the line). Use truth tables to show that this is indeed the case.

$$\textbf{MP}: \frac{P \qquad\qquad P \rightarrow Q}{Q}$$

2. Here is a small Prolog program.

```
p :- q, r.
q :- s.
r :- t.
t :- r.
s.
```

   (a) What are the corresponding logical statements in the declarative reading of the program, using standard logical notation?

   (b) A backchain inference for propositional definite clause problems can be described as follows.

```
function BC_Query ( Prog, Goals ) returns boolean
  inputs  Prog, list of prop definite clauses
          Goals, list of (atomic) goals

  if Goals is empty  then return TRUE
  else { let G :: GRest = Goals;
         for each F1 & ... & Fn -> G  in Prog
              {  NewGoals = append ( [F1, ... Fn], GRest );
                 BC_Query ( Prog, NewGoals )
              }
       }
  return FALSE
```

   The matching clauses at the "for each" step are searched in the order in which they appear in the program. A unit clause is treated as a head atom, with an empty list as the body.

   Draw the search tree that the standard Prolog search explores for the program above, by giving the first few levels of the tree, for the initial query ?- p.

(c) The Prolog inference procedure is *incomplete*. Say what this means. Does the example discussed above give an instance of this incompleteness?

(d) What is a *decision procedure* for the problem of deciding if a given atom is a logical consequence of a given $\mathcal{S}$? Sketch how the Prolog depth-first top-down inference procedure could be adapted to provide a decision procedure for inference from propositional definite clauses.

3. Recall the fixed point approach to characterise the logical consequences of a set of propositional definite clauses $\mathcal{S}$. Let A be the atomic propositions appearing anywhere in the formulas in $\mathcal{S}$, and $Base$ be the atoms that appear as a clause on their own in the program (unit clauses). The function $f : \mathcal{P}(A) \to \mathcal{P}(A)$ is defined as:

$$\begin{aligned} f(Y) &= Y \cup Base \\ &\quad \cup \{\, a \in A : (s_1 \wedge \cdots \wedge s_n \to a) \text{ is in } \mathcal{S}, s_1 \in Y, \ldots, s_n \in Y \,\} \end{aligned}$$

(a) Work out $f(\{\ \}), f(f(\{\ \})), f(f(f(\{\ \})))$ for the program in question 2. What is the fixed point of this f?

(b) What does it mean for a function $f : \mathcal{P}(A) \to \mathcal{P}(A)$ to be monotone? Show that for any set of definite clauses $\mathcal{S}$, the corresponding f is monotone.

4. Suppose that we extend the language in the programs to allow negated atoms to appear in clauses (this is standard logical negation with the usual truth table, **not** Prolog negation, which is something else).

Show that p is a logical consequence of the following program, translated as above into standard logic: p :- $\neg$ p.

We will look at Prolog's treatment of negation later.