3ba2 Artificial Intelligence

Lab: November 25, 2004

1 Logical consequence bottom-up

Recall from the lecture of Wednesday, November 17th[1] that we can

(i) encode a propositional knowledge base as a list of lists — e.g.

```
a  :- b,c.
a  :- d.
b  :- c.
c  :- d.
d.
```

as `[[a,b,c],[a,d],[b,c],[c,d],[d]]`

(ii) define a graph from such a knowledge base, where a node is a list of facts to prove (to-do list/agenda) and

```
arc([H|T],X,KB) :- member([H|B],KB), append(B,T,X).
```

(iii) check if a fact `X` follows from a knowledge base `KB` through predicates

```
seek([X],KB)
search([[X]],KB)
```

that search the graph in (ii) top-down from start node `[X]` to goal node `[]`.

Your task is to check if `X` follows from `KB` by reversing the graph above, searching bottom-up from start node `[]` to some goal node `N` such that

```
member(X,N).
```

Question. Which is preferable in this case: breadth-first or depth-first?

2 Finite-state machines in Prolog

A finite state machine `M` can be regarded as a pair (`Trans, Final`) where

- `Trans` is a list of triples `[Q1,X,Q2]` such that at state `Q1`, `M` moves to `Q2` if it sees the symbol `X`

and

- `Final` is a list of `M`'s final (i.e. accepting) states.

[1] See `www.cs.tcd.ie/Tim.Fernando/3ba2/nov17.pl`, which I hope to move to the webpage Mike Brady set up, as soon as I am permitted to do so.

Let us agree to name `M`'s initial state `q0`, and to encode strings as lists (e.g. 100 as `[1,0,0]`).

Your task is to define a 3-ary predicate `accept` such that

`accept(Trans,Final,String)` iff (`Trans,Final`) accepts `String`.

That is, write a knowledge base that a Prolog interpreter can consult to answer queries such as

```
| ?- accept([[q0,0,q1],[q0,1,q1],[q1,0,q0],[q1,1,q0]], [q1], [1,0,0]).
```

Be prepared to explain your knowledge base (document it with comments), and to demonstrate it.