CS 367 Tutorial
20 October 2008
Week 12 (tutorial #10)
Carl Schultz

Prolog

declarations
(page 71 of prolog manual)

- lines that start with ":-" are declarations
- used to tell prolog to treat certain predicates in a special way

```
:- multifile derived/1
```

- …means that if more clauses are loaded from other files for the predicate "derived", the new clauses will be added to the old ones (rather than replace them)

```
:- dynamic derived/1
```

- …means that other predicates might inspect, add or delete some of the "derived" clauses

modifying the database
(page 152 of manual)

- dynamic predicates can be changed and inspected at runtime by other clauses
- `assert` and `retract` are used to add and remove clauses

```
:- dynamic need/1.
raining :- assert(need(umbrella)).
sunny :- retract(need(umbrella)).

| ?- need(X).
no
| ?- raining.
yes
| ?- need(X).
X = umbrella ? ;
no
| ?- sunny.
yes
| ?- need(X).
no
```

- `functor` is used to match a predicate to a name and arity

```
| ?- functor(foo(a,b), N, A).
N = foo,
A = 2
| ?- functor(X, foo, 2).
X = foo(_A,_B)
```

- you will need to use `assert` your operators and heuristic function so that they can be inserted into the database at runtime

idaStar.pl

f-bounded (f-limited) search is the main relation

`fbsearch / 5`

(review IDA powerpoint week 11)*

1. check if node is a goal node
2. check if the F value (path cost + heuristic value) is less than or equal to the bound → if yes, then add children nodes to the frontier (i.e. nodes to visit), and visit one of these children (depth-first search with recursively call to fbsearch/5)
3. check if F value is greater than bound → if yes, then record this F if it's the smallest F over the bound so far (i.e. keeping track of the minimum F over the bound, preparing for the next change in bound)
4. F value of all nodes are over the bound → start again, but increase the bound

For this to work, you need to define:
Domain definitions:

- neighbors(State, Neighbors)
- cost(State, Neighbor, ArcCost)

Problem definition:

- isGoal(State)

Search definition:

- h(State, Goal, HeuristicValue)

progressionPlanning.pl

`neighbors(State, Neighbors)`

- returns children nodes (neighbours) of given node (State)
- collects neighbours using built in "setof" predicate (look this up in the manual, also "^" and "bagof")

1. get applicable operations (e.g. move) → do this by testing whether an operations preconditions are satisfied by the currect state
2. apply the operation to get the new state (the neighbour) → this basically means modifying the *fluents* (statics don't change between states)
   - remove fluents that operation has made false (e.g. it's no longer true that "at(warehouseman, pos(2,3))" so remove it)
   - add fluents that operation has made true (e.g. "at(warehouseman, pos(3,3))")

note: `stateFluents(State, StateFluents)` is meta-level → it basically checks each predicate in State to see if it's a fluent, and if it is, it adds it to StateFluents list