CIS471/571
Artificial Intelligence
Winter 2007

Midterm

Constraint Satisfaction

1. In a sentence describe the effect of constraint propagation during search for solution. (5)

Constraint propagation removes possible values from unassigned variable ranges, reducing the size of the remaining search space (or search branching factor).

2. Name a local search method for solving constraint satisfaction/optimization problems. (5)

hill climbing, simulated annealing, genetic algorithms

3. What is the MRV (most restricted variable) heuristic? (5)

That is selecting the variable with fewest number of remaining possible values, again to try to reduce the branching factor of the search.

Path Problem Solving

A search space for a path problem is a set of states interconnected directed arcs representing allowable operators.

1. Describe a search method that, even when given a finite search space, might produce an infinite (unending) search. (5)

Searching without removing duplicate states in a search space with directed cycles.

2. What types of search spaces would always result in finite searches? (5)

Finite, tree-structured (directed acyclic) search spaces.
Path Problem Solving

3. Describe the iterative deepening search algorithm. Why is it preferred when we don’t know much about the state space? (10)

Searching in depth-first fashion with increasing depth limits. It takes less than twice the time but space that only increases linearly with depth.

4. What is a heuristic function (what does it estimate)? What is an admissible heuristic function? Why is it valuable to have an admissible heuristic function? (10)

A heuristic function estimates the cost to reach a solution state from a given state. An admissible heuristic function is one that never overestimates the cost to reach a goal state. Using an admissible heuristic function with A* guarantees a minimum cost solution.

5. Consider a context where blocks labeled by letters (A, B, C, D, .....), are placed in 3 stacks (1, 2, 3). Actions are to pick up a block from the top of some stack or place a block that is in the hand on the top of a stack. Assume the state predicates:
   onTop(<block, <stack>), inHand(<block>), emptyHand()
   on(<block>, <block>), emptyStack(<stack>)

Define the action PICK-UP-TOP(<stack>) which picks up a block from a stack of more than one block in a STRIPS-like format: (10)

```
PICK-UP-TOP(x):
  PRE: onTop(y, x)
  on(y, z)
  emptyHand()
  ADD: inHand(y)
  onTop(z, x)
  DEL: emptyHand()
  onTop(y, x)
```
Game Playing

1. What is a static evaluation function? (5)

*It is a function applied to a game state, estimating how good the state is.*

2. What is the horizon effect? Why does it exist for most game playing programs? How can we try to minimize its effect? (10)

*This reflects the error in a static evaluation function's value due to the arbitrary cut off of search. There is always somewhat of a horizon effect due to the fact that search must stop short of a final state in a large game tree. One can try to minimize its impact by quiescence analysis, stopping search in quiet positions.*

Knowledge Representation

1. Prove that the clauses {1.(red, ~green) 2. (green, ~blue) 3. (blue, yellow, orange) 4. (~orange, ~pink) 5. (~pink, ~red) 6. (pink)} logically entail yellow by resolution. (10)

```
add (~yellow) 7.
6,4 -> (~orange) 8.
8, 3 -> (blue, yellow) 9
9, 7 -> (blue) 10.
6, 5 -> (~red) 11.
11,1 -> (~green) 12.
12,2 -> (~blue) 13.
13, 10 -> () qed
```

2. Are the following two first-order clauses resolvable and, if so, what is the most general resolvent. (5)

```
[~P(a, x, y), Q(y, x, b)]  [R(x, a, b), P(x, y, b)]

can not be resolved, as can not unify argument list of P and ~P
```

3. Give an example of a statement in the situation calculus. What is the situation calculus used to represent? (5)

*Holds(on(A, B), s). Situation calculus has been used to represent planning domains in a logical framework... to reason about actions and situations and their implications.*
4. Consider reasoning about blocks labeled by letters on numbered stacks. Given the following knowledge base:

Facts: \( \text{AtBottom}(A, 1), \text{On}(B, A), \text{Clear}(B), \text{AtBottom}(C, 2), \text{On}(D, C), \text{On}(E, D), \text{Clear}(E) \)

Rules: 
- R1: \( \text{On}(x, y) \Rightarrow \text{Above}(x, y) \)
- R2: \( \text{On}(x, y) \) and \( \text{Above}(y, z) \) \( \Rightarrow \) \( \text{Above}(x, z) \)
- R3: \( \text{Clear}(x) \) and \( \text{Above}(x, y) \) and \( \text{AtBottom}(y, z) \) \( \Rightarrow \) \( \text{OnTop}(x, z) \)

Give a trace of Backward Reasoning (Backward Chaining) for the goal \( \text{OnTop}(B, x) \) (i.e., that B is on top of some pile).

A trace can be presented as the sequence of values on the goal stack. Push rule conditions on the stack so first condition is at the top of the stack.

\[
\begin{align*}
[\text{OnTop}(B, x)] & \quad \text{R3} \\
[\text{Clear}(B), \text{Above}(B, v1), \text{AtBottom}(v1, v2)] & \quad \text{Facts} \\
[\text{Above}(B, v1), \text{AtBottom}(v1, v2)] & \quad \text{R1} \\
[\text{On}(B, v1), \text{AtBottom}(v1, v2)] & \quad \text{Facts} \\
[\text{AtBottom}(A, v2)] & \quad \text{Facts} \\
[] & \quad \text{Facts}
\end{align*}
\]