Midterm

CSC 242

3 March 2005

*Write your **NAME** legibly on the bluebook. Work all problems. You may use three double-sided pages of notes. Please hand your notes in with your bluebook. The best strategy is not to spend more than the indicated time on any question (minutes = points).*

1. Heuristic Search: 15 Min. Suppose you want to use AI techniques to create $N \times N$ panmagic squares. That is, the elements are the integers between 1 and N^2 inclusive, and the sum along all rows and all columns and the main diagonals is the same, like this one for $N = 4$.

```
13  2 16  3
12  7  9  6
 1 14  4 15
 8 11  5 10
```

(There are many varieties of magic squares... see `http://www.pse.che.tohoku.ac.jp/~msuzuki/MagicSquare.html`, for instance. Also there are deterministic ways to produce them Also I bet not every NxN panmagic square exists. But we digress...).

In as much technical detail as you have time for, say how you would set up this problem as:
(A) (5 Min.): A constrained search problem (CSP).
(B) (5 Min.): A heuristic search problem using the A* algorithm.
(C) (5 Min.): A hill-climbing or simulated annealing search problem.

Answer: It helps to figure out what the magic sum is... easy.

Most natural as CSP. The square starts out empty and numbers are placed in it one by one. The constraints come from the definition of panmagic. The variables in the problem are the square's elements. Thus all the row's elements must total 34, all elements must be different, etc. etc. Our operators are the assignment of an unassigned (due to constraints) integer value to a square position (variable). Simple DFS would be one way to get a solution, but backtracking search is more eficient... it checks all constraints whenever any variable is assigned. Even better is forward checking... this makes sure other variables can achieve legal values given the current assignment, and eliminates possibilities that cannot. This detects failure and narrows search both. Constraint propagation could work too, in which changes in the possible values of each variable recursively affect the values of other variables until the whole system settles down into a so-far consistent state.

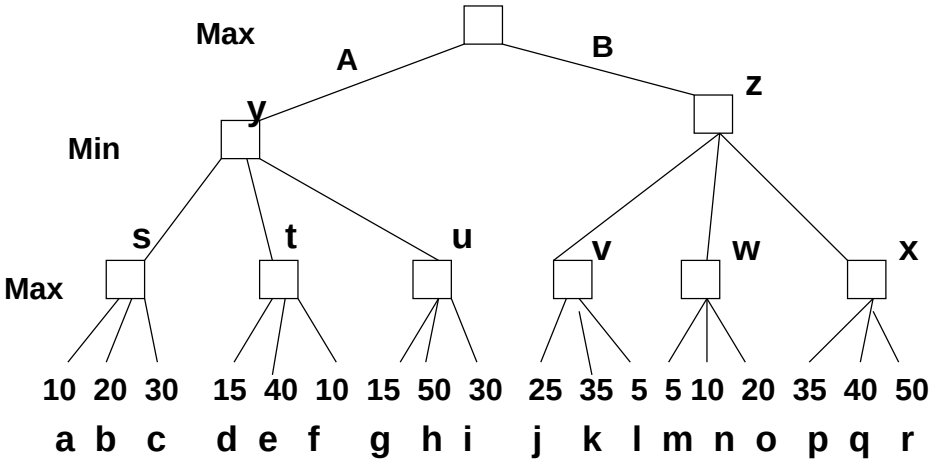
Normally I think of setting up this problem for heuristic search with the square empty and an operation of placing a number. Equally good, I think, is starting with a full square and swapping.

For A*, I wanted to know your h and g functions. f = g + h, g is how many swaps or numbers placed, h is maybe how many rows, cols, diags are not equal to the magic sum (34 for 4 x 4). But I wonder if that is admissible. A good argument for or against admissibility of your heuristic is a nice finishing touch to your answer.

Ssimulated annealing or hill-climbing: similar. Important to say you start with a "complete state", that is a full square, with swaps being the op, and goodness figure is maybe the h above, or something like the absolute value of the average difference of the sums from what they should be...

2. Minimax and $\alpha - \beta$: 15 Min.

(A) (5 Min.): Consider the following tree from the usual sort of zero-sum, two person game of perfect information, in which the indicated static scores are all from the first player (MAX)'s point of view. What move (A or B) should MAX choose and why?



(B) (5 Min.): Which nodes (interior and leaf, a–z) would not be examined if alpha-beta pruning is used?

(C) (5 Min.): Suppose the move generator produces moves so that the static evaluation of positions is in the optimal order for $\alpha - \beta$ pruning? Draw the resulting tree and indicate which nodes would not be examined.

Answer: (A) Max chooses A since its backed up value is 30 vs. 20 for B

(B) f, i, x,p,q,r

(C) -'s not examined: positions generated in order of goodness: numbers show order if all the nodes were actually expanded. So in this case these nodes are not expanded: $d, f, i, g, v, (j, k, l), x, (p, q, r)$.

In the left subtree you can think of Max knowing Min isn't going to choose t or u, since that would let Max do better than 30. On the right, Max notices Min can force a 20 on him by going to w, so Max abandons the entire z subtree. Max has to look at all the alternatives under w to make sure there's nothing good lurking there...there's a bit of a philosophical puzzle here since we're pretending we generate these nodes in order but then pretend we don't know it! – see me if you're puzzled.

```
                    top
          y                         z

  s       t     u           w       -       -
```

```
c  b  a    e  -  -    h  -  -    o  n  m    -  -  -    -  -  -
30 20 10   40 15 10   50 30 15   20 10 05   35 25 05   50 40 35
```

3. Resolution Proof: 15 Min. Given these premises:
1. Anyone who procrastinates is sluggish.
2. Everybody has a friend who procrastinates.

Express them in FOPC, put them into clause form, and use resolution to answer the question: Is anyone sluggish and if so, who?

Answer:

$$\forall x(P(x) \Rightarrow S(x))$$

$$\forall x \exists y(F(x,y) \wedge P(y))$$

So we get clauses

$$\neg P(x) \vee S(x)(1)$$

$$F(x, f(x))(2)$$

$$P(f(x))(3)$$

Assert that no one is sluggish:

$$\neg S(x)(4)$$

Unify (3) and (1) , and the result with (4) (or the other order) and we find that if you give me any x, I'll give you good old $f(x)$, the individual generated by the Skolem function, in short it's x's laid-back friend who procrastinates and who is sluggish. This is the lackadaisical individual who gives us the substitution that allows the derivation of the null clause. Another correc way to axiomatize this situation would be to use another assertion to the effect that someone really does exist. This existence will skolemize to some individual A, and it will then be A's friend who puts things off until the last minute.

4. Unification: 15 Min.

You are given the clauses:

$$L(x, y, g(A, y), D)$$

$$L(z, C, g(w, u), v)$$

with variables u, v, w, x, y, z; constants A, C, and D; function g and predicate L.

(A) (5 Min.) Find the most general unifying substitutions and show the result of unifying the clauses.

Answer:

(I'm using | for the "substitution backslash").

$w \mid A, y \mid C, u \mid C, v \mid D, x \mid z$ (or vice versa) to give (e.g.)

$L(x, C, g(A, C), D)$.

(B) (5 Min.) Show a less general unifying substitutions and show the result of unifying the clauses.

Answer: use $x \mid E$ and $z \mid E$, instead of $x \mid z$, say. The idea is to have less generality by having fewer variables and more individuals result from the unification.

(C) (5 Min.) How would your answer to a) change if the clauses were

$$L(x, y, g(A, y), D)$$

$$L(y, C, g(x, u), z)?$$

Answer: It wouldn't. The y in one expression is a totally different variable from the y in the other, and the normal unification rules work just fine if they're done right. Common practice is to relabel variables so we don't get mixed up, however.

5. Natural Language: 15 Min.

You give your Quagent Curly the order: "Curly, GoTo the door with the gold."

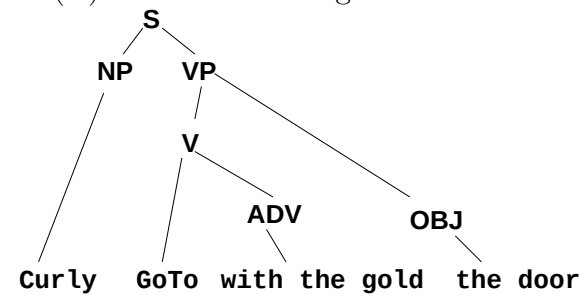
(A) (5 Min.) Curly has cause to be a bit puzzled. He creates two different parse trees using his simplest English-language phrase structure grammar. Give an idea of what they look like; that is, show how the two different ways to parse the sentence result in different trees.

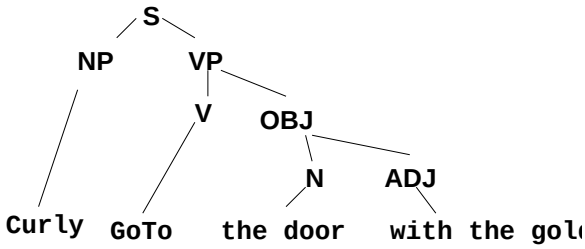
(B) (5 Min.) Does the sentence have referential ambiguity? semantical ambiguity? lexical ambiguity? syntactic ambiguity? Say why or why not in each case.

(C) (5 Min.) What approaches might Curly use to disambiguate the sentence: 1) just by itself, 2) in the context of an ongoing dialog, and 3) just by itself but using his Quagent senses.

Ans.

(A) This sort of thing is all I wanted for the "parse tree":





(B) Syntactic yes (that's the parse tree(s), thus semantic since the meanings are different owing to the different role of the gold; Referential yes: we don't know what door or what gold we're talking about here. lexical no, we know what all the words mean and they seem unambiguous in this case.

(C) 1) By itself, rather a tough one. I meant "by itself" – no sensing, no introspection, no knowledge of map, nothing but the order. Semantic grammars to the rescue, though! If there's only one parse there's no ambiguity. High-level models and probabilities of gold being at doors would help. 2) In a dialog context, I guess Curly could ask what was wanted. That wasn't in the spirit of the question, but I allowed it: turns out not to be that easy to do right. Anyway, without that approach. one could search for bindings for the door and gold that would restrict the gold to something playing a being-carried role by Curly or a door-discriminating role. Simpler ideas would be to use recency of reference to figure out the bindings: If "PickUp the gold." immediately preceded our S. we would bind to that. 3) Curly can look around using RAYS or RADIUS, see if any doors have gold close to them, can examine his inventory to see if he has gold on him already, if there is gold nearby that he can PickUp and only one door he can GoTo, etc. This last kind of disambiguation is clearly useful but is not so common in current NLU systems. Some of the issues are deep and Randal Nelson is working on them here at UR if you want to get involved.