

Quiz 1 Review Solutions
October 8, 2005

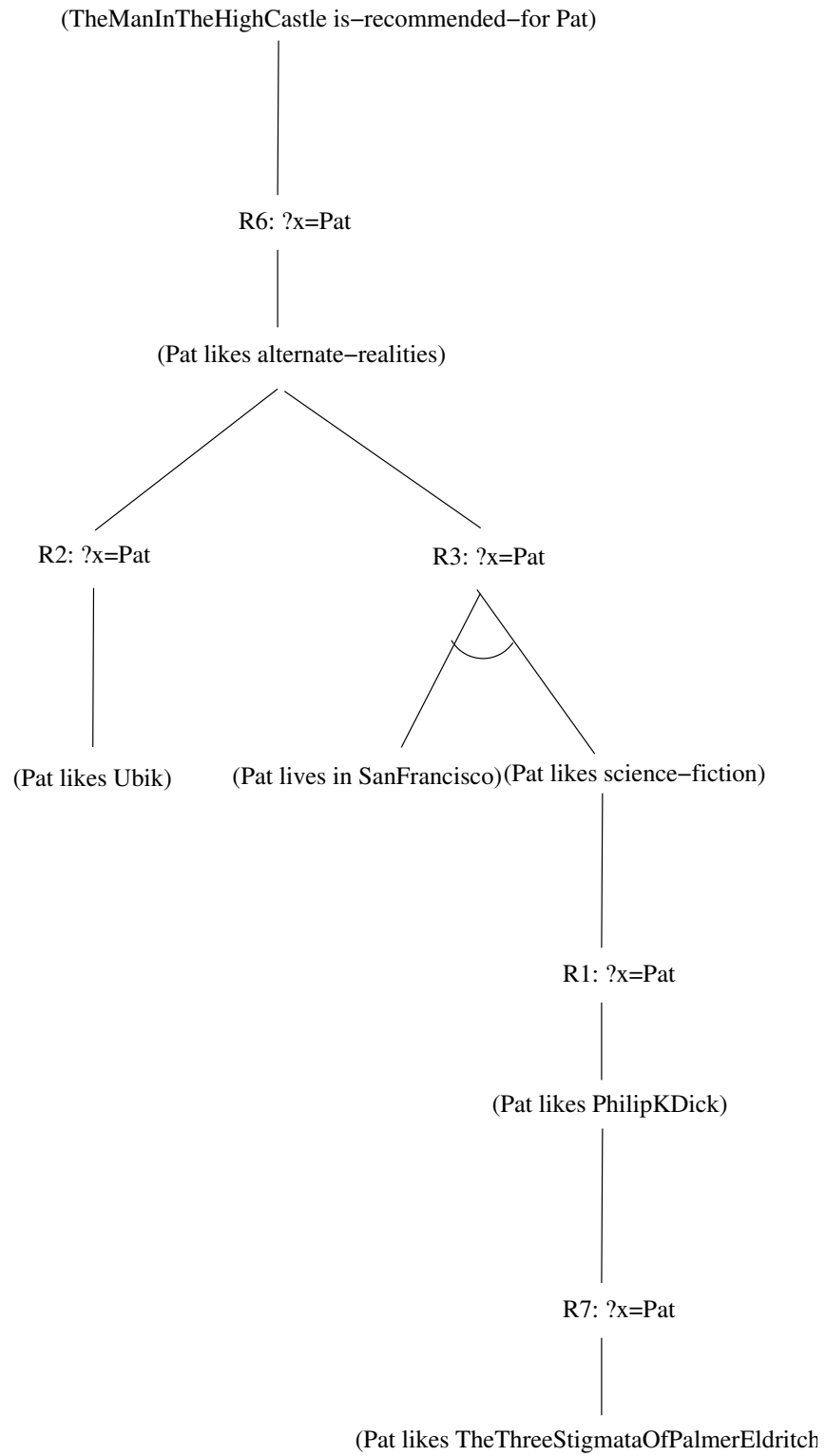
1. Author Rules Problem Solution ¹

(a)

Step	Triggered Rule(s)	Rule Instance Binding(s)	Rule Fired	Database Assertion(s) Added
1	R1	?x = Jane	R1	(Jane likes science-fiction)
	R4	?x = Max		
2	R3	?x = Jane	R3	(Jane likes alternate-realities)
	R4	?x = Max		
3	R4	?x = Max		(Max likes politics)
	R6	?x = Jane		
4	R5	?x = Max	R5	(ThePenultimateTruth is-recommended-for Max)
	R6	?x = Jane		
5	R6	?x = Jane	R6	(TheManInTheHighCastle is-recommended-for Jane)

¹This rules problem solution is due to Kimberle Koile.

(b)



2. Robot Rules Problem Solution ²

(a) Forward Chaining

- i.
- | Step | Room robot is in |
|------|------------------|
| 1. | rm1 |
| 2. | rm2 |
| 3. | rm5 |
| 4. | rm6 |
| 5. | rm1 |
| 6. | rm2 |

The robot loops because the GO rule is always triggered and chosen to fire since it is first, and the (door rm2 rm5) assertion is found before the (door rm2 rm3) assertion. So the robot never gets to the other rooms.

- ii. Adding the SYM rule would not change anything if it is placed at the end of the rules because the GO rule would fire first.
- iii. Adding the SYM rule before the GO rule would cause more door assertions to be added to the database. If we assume the new assertions are added after the existing ones, the behavior of the system does not change; *i.e.*, the robot still loops.

- iv.
- (door rm1 rm2)
 - (door rm4 rm3)
 - (door rm2 rm3)
 - (door rm2 rm5)
 - (door rm5 rm6)
 - (door rm6 rm1)

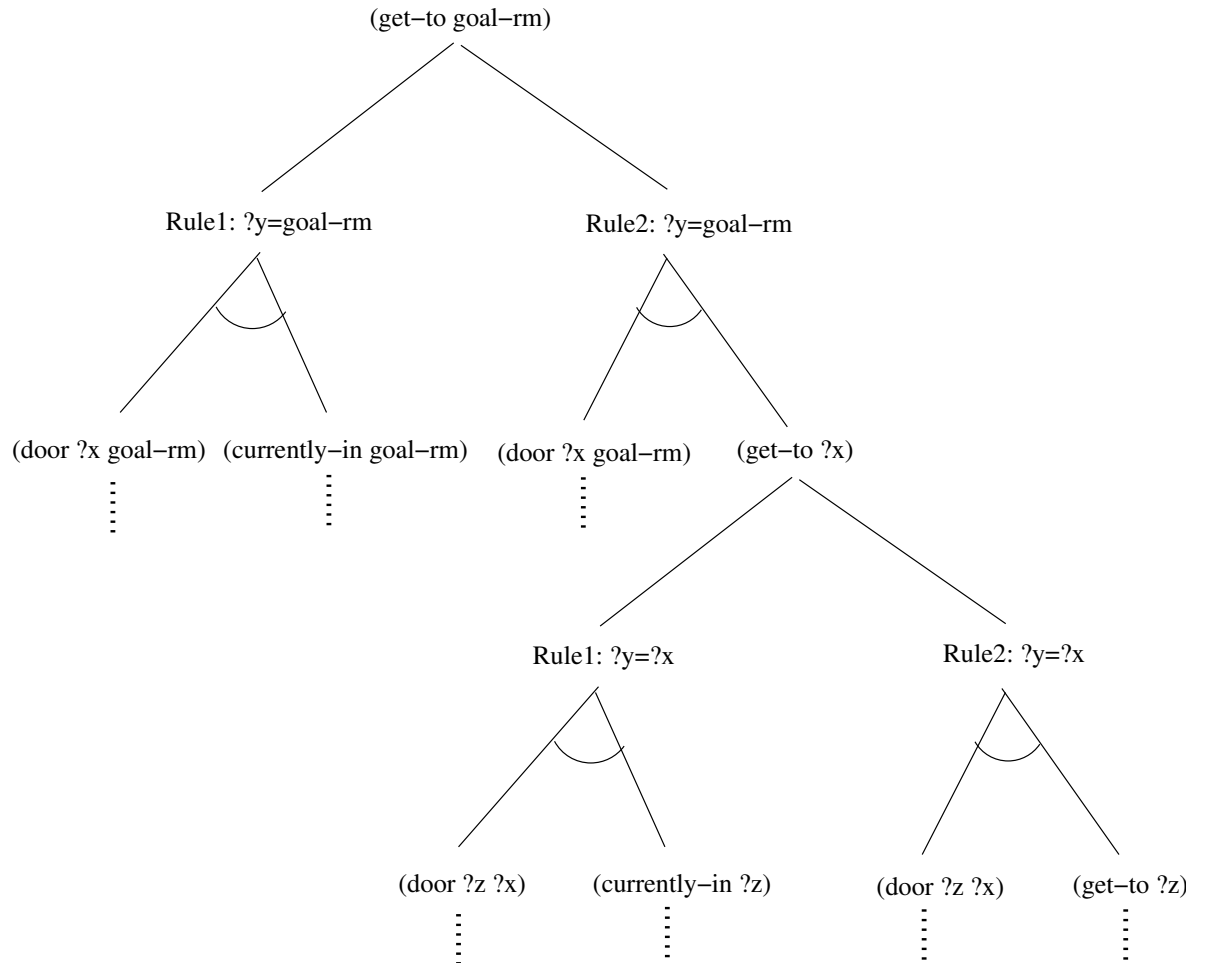
Step	Room robot is in (without SYM)	Room robot is in (with SYM)
1.	rm1	rm1
2.	rm2	rm2
3.	rm3	rm3
4.	rm4	
5.	rm3	
6.	rm4	

The robot loops. To get the robot to stop, the STOP rule must be before the GO rule.

²This rules problem solution is due to Kimberle Koile.

(b) Backward Chaining

i.



The system starts by trying to prove the (get-to goal-rm) assertion. It doesn't find an assertion in the database that matches it, so it looks for rules that might prove (get-to ?x) with a binding of ?x=goal-rm. It finds Rule1 and Rule2. Since it finds Rule1 first (and it is using rule ordering for conflict resolution), it tries to match the antecedents in that rule, looking in the database for assertions that match (door ?x goal-rm) and (currently-in ?x). It finds the assertions (door rm1 goal-rm), so binds ?x to rm1, then tries to prove the second antecedent (currently-in rm1). It doesn't find that assertion in the database, nor a rule that proves that assertion, so it queries the user to find out if the assertion is true. The answer is no, so it continues. (Note: some systems instead of querying the user may just assume that if an assertion can't be found in the database or proven by rules, that it is assumed to be false. Here we're assuming the user is queried instead.) It then looks in the database for another matching assertion, finds (door rm2 goal-rm), then tries to prove (currently-in rm2). It can't find that assertion in the database, nor a rule that proves it, so it queries the user. The answer is no, so the system continues looking for other matching assertions. It continues in the same fashion, finding rm3

and rm4. Since none of these bindings for $?x$ prove the goal assertion (i.e. the root node here, which is also sometimes called a hypothesis), Rule1 fails, and the system tries to prove Rule2. It again has a conjunction, so it looks for bindings for the first antecedent (door $?x$ goal-rm), finds (door rm1 goal-rm) and binds $?x$ to rm1. It then tries to prove the second antecedent, (get-to $?x$) with $?x$ also bound to rm1. It then finds that Rule1 and Rule2 could prove that assertion, so tries Rule1 first (because it is using rule ordering for conflict resolution). It now looks for assertions matching (door $?z$ rm1), finds (door rm5 rm1), binds $?z$ to rm5, then tries to prove (currently-in rm5). That fails because it's not in the database and a query to the user yields no, so it continues, finding (door rm6 rm1). It attempts to then prove (currently-in rm6), and this fails. Finding no more assertions that match (door $?z$ rm1), it tries Rule2 to prove (get-to rm1). Etc....

- ii. Backward chaining is not appropriate when there are many paths leading away from the goal. Forward chaining often works much better in these cases.

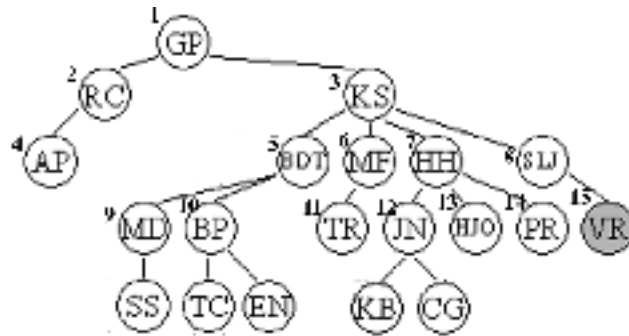
3. Using search for something other than a map! ³

- (a) There are several possible answers, but one that works is Guy Pearce → Kevin Spacey → Helen Hunt → Jack Nicholson → Kevin Bacon.
- (b)



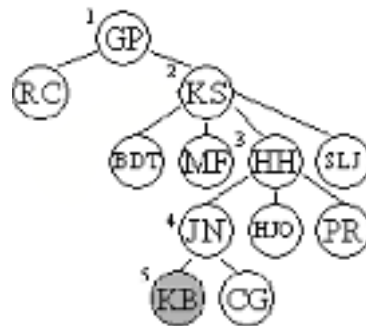
³This problem solution is originally due to Kimberle Koile — used here with minor modifications.

(c)



(d) Number of actors listed as co-stars.

(e)



(f) If you had a bigger database with more actors listed and more co-stars for each one, using beam search might help limit the size of the problem from getting too huge, and the same heuristic could be used. But remember: beam search is not guaranteed to find a path.

4. *A** to the rescue! Solution ⁴

Queue	Extended list	Next node to extend	Not extended
1. (1 S)	()	(1 S)	extended
2. (13 A S) (14 B S) (17 C S)	S	(13 A S)	
3. (16 B A S) (20 D A S) (13 E A S) (17 F A S) (14 B S) (17 C S)	S, A	(13 E A S)	
4. (21 D E A S) (23 B E A S) (13 G E A S) (19 F E A S) (16 B A S) (20 D A S) (17 F A S) (14 B S) (17 C S)	S, A, E	done	

⁴This optimal search solution is due to Kimberle Koile—used here with minor modifications.

Node extension order: **S A E**

Path: **S A E G** (No, the robot did not get there in time; but he made skis out of left behind metal and was happy.)

5. The question is essentially asking you for the largest value of c that keeps the heuristics admissible (*i.e.*, does not overestimate the cost of an optimal path to the goal). The answer is

$$c = \min(3, 2a + 1, 2 + 3a/2, 7a/2),$$

which results from analyzing the cost of the paths to the goal from the nodes with heuristic values involving c . Note that the conditions for the heuristic value $c/2$ are not needed since they are always weaker than those for c . Eliminating dominated terms, we get the simplified expression

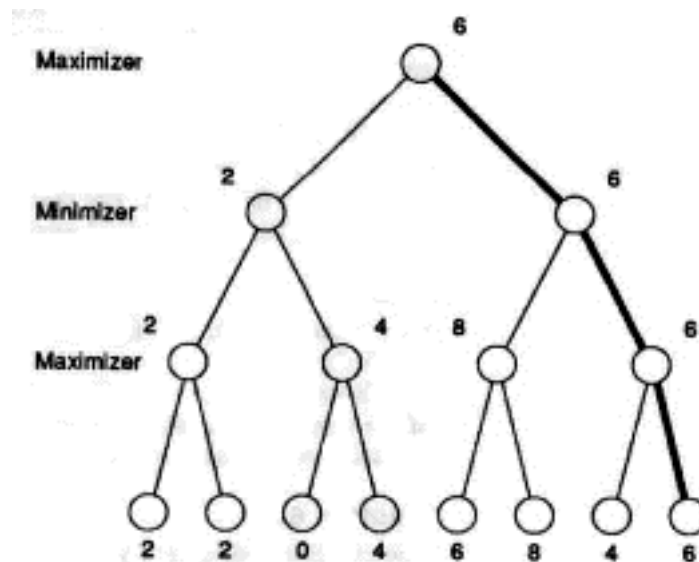
$$c = \min(3, 2a + 1, 7a/2),$$

which can alternatively be expressed as

$$c = \begin{cases} 3 & \text{if } 1 \leq a, \\ 2a + 1 & \text{if } 2/3 \leq a < 1, \\ 7a/2 & \text{if } 0 \leq a < 2/3. \end{cases}$$

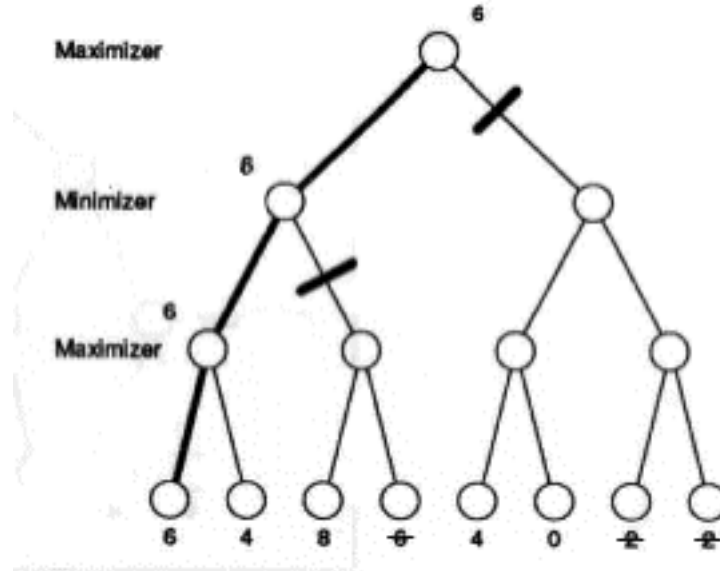
6. (Solution to Exercises 6.3 and 6.4 from the book)⁵

(a)



⁵The first two parts of this search in games problem solution are from the book *Artificial Intelligence* by Patrick Winston. The last part is due to Kimberle Koile.

(b)

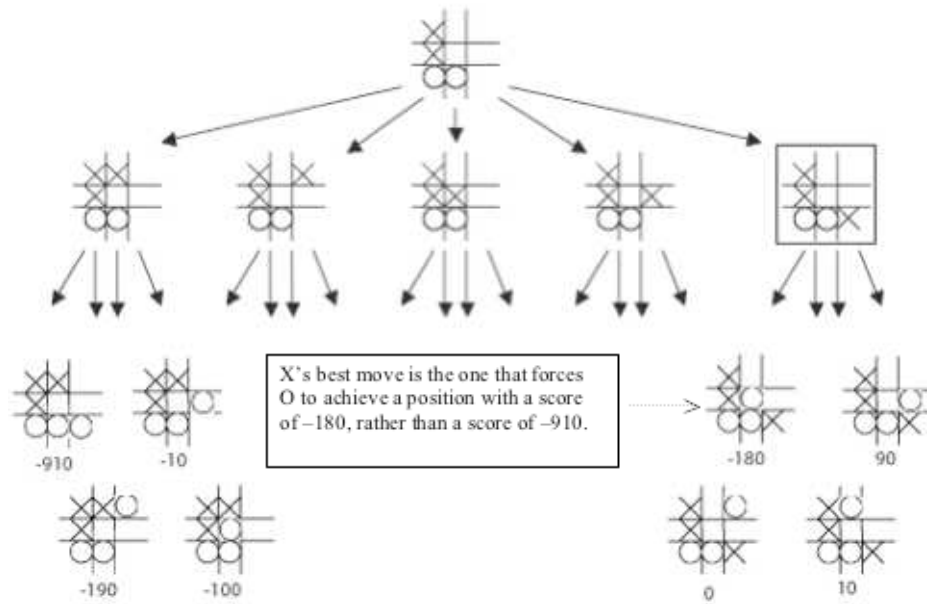


- (c) If the evaluated nodes are ordered in the manner described below, then you get maximal Alpha-Beta cutoff; the opposite order gets no Alpha-Beta cutoff. For the game trees we've been looking at (*i.e.*, with the bottom row containing the evaluated nodes, and successively higher layers of nodes alternating between minimizer and maximizer, or maximizer and minimizer):

If penultimate level of the tree is maximizer level, you get maximal cutoff if descendents of each node in that level are ordered from left to right, max value to min value. If penultimate level of tree is minimizer level, you get maximal cutoff if descendents of each node in that level are ordered from left to right, min value to max value.

Note that since the goal of Alpha-Beta pruning is to cut down on the number of nodes that have to be evaluated, game programs don't actually sort nodes. Two observations that can be made: (1) the performance of Alpha-Beta is variable and can't be counted on to always outperform Minimax; (2) a move generator could attempt to produce new configurations in a sorted order.

7. Tic-Tac-Toe Solution ⁶



⁶This search in games problem solution is due to Kimberle Koile.