CS 471 HW 2

Ankur Dhoot

February 22, 2017

Q1

(a)

$$\begin{tikzcd}
 & & & & \boxed{(7,12)} \arrow[dlll] \arrow[d] \arrow[drrr] & & & & \\
 & \boxed{(7,12)} \arrow[dl] \arrow[dr] & & & \boxed{(3,14)} \arrow[dl] \arrow[dr] & & & \boxed{(1,9)} \arrow[dl] \arrow[dr] & \\
\boxed{(11,8)} & & \boxed{(7,12)} & \boxed{(12,5)} & & \boxed{(3,14)} & \boxed{(1,9)} & & \boxed{(5,-2)}
\end{tikzcd}$$

(b) Suppose we're at a node corresponding to Player B. We have values α and β corresponding to the value of the best choice for Player A and Player B, respectively, that we have found so far. We are given that $|U_A + U_B| \leq k$. Suppose we explore from the current node and find that $U_B \geq k - \alpha$. This implies $U_A \leq \alpha$ by our given constraint. Thus, we don't need to further explore paths from the current node and the node can be pruned. The analogous inequalities hold if we're at a node corresponding to Player A.

Q2

Q3

We're given the following:

$A \wedge B \Rightarrow D$

$Q \wedge \neg R \Rightarrow A$

$\neg Q \vee \neg B \vee \neg R$

$P \Rightarrow Q$

$P \Rightarrow B$

$B \Rightarrow P$

B

(a) We'll prove D using forward chaining where we check to see if premises are satisfied and if so, add the conclusion to our inferred set:

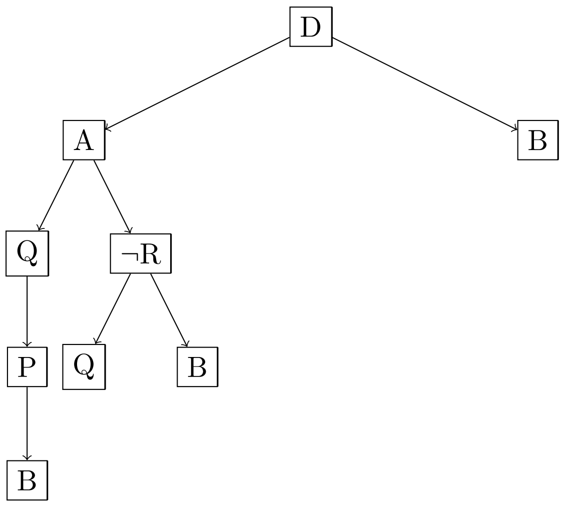
First note that we'll rewrite $\neg Q \vee \neg B \vee \neg R$ as $Q \wedge B \Rightarrow \neg R$.

B
$B \Rightarrow P$
$P \Rightarrow Q$
$Q \wedge B \Rightarrow \neg R$
$Q \wedge \neg R \Rightarrow A$
$A \wedge B \Rightarrow D$

(b) Using backward chaining we'll identify the premises we must satisfy recursively:

D
$A \wedge B \Rightarrow D$
$Q \wedge \neg R \Rightarrow A$
$P \Rightarrow Q$
$B \Rightarrow P$
B
$Q \wedge B \Rightarrow \neg R$

In search tree form, read left to right as in DFS (similar to Figure 9.7 in RN):


$$\begin{tikzcd}
 & & \boxed{D} \arrow[dll] \arrow[dr] & \\
\boxed{A} \arrow[d] \arrow[dr] & & & \boxed{B} \\
\boxed{Q} \arrow[d] & \boxed{\neg R} \arrow[d] \arrow[dr] & & \\
\boxed{P} \arrow[d] & \boxed{Q} & \boxed{B} & \\
\boxed{B} & & &
\end{tikzcd}$$

Q4

(a) 1. Captain(Enterprise, Kirk)

2. $\exists x$ $\forall y$ HasDisease(y,x) $\Rightarrow$ Hungry(y)
3. $\forall x$ OnShip(Enterprise, x) $\Rightarrow$ $\neg$ Diseased(x)
4. Pizzas(Kirk)
5. $\forall x$ Pizzas(x) $\Rightarrow$ Hungry(x)

6. $\forall x$ [$\exists y$ Captain(y, x)] $\Rightarrow$ OnShip(x, y)
7. Hungry(Kirk) $\Leftrightarrow$ Sick(Kirk)

(b) 1. Captain(Enterprise, Kirk)
2. $\neg$ HasDisease(y, C1) $\vee$ Hungry(y) (Skolem Constant C1)
3. $\neg$ OnShip(Enterprise, x) $\vee$ $\neg$ Diseased(x)
4. Pizzas(Kirk)
5. $\neg$ Pizzas(x) $\vee$ Hungry(x)
6. $\forall x$ $\neg$[$\exists y$ Captain(y, x)] $\vee$ OnShip(x, y)
$\forall x$ [$\forall y$ $\neg$ Captain(y, x)] $\vee$ OnShip(x, y) (using rules for negated quantifiers)
$\neg$ Captain(y, x) $\vee$ OnShip(x, y) (after removing universal quantifiers)
7. ($\neg$ Hungry(Kirk) $\vee$ Sick(Kirk)) $\wedge$ ($\neg$ Sick(Kirk) $\vee$ Hungry(Kirk))

(c) We'll add a couple common sense sentences: $\forall x$ Sick(x) $\Rightarrow$ Diseased(x) which converts to $\neg$ Sick(x) $\vee$ Diseased(x)
1. Pizzas(Kirk)
$\neg$ Pizzas(x) $\vee$ Hungry(x)
$\therefore$ Hungry(Kirk)

2. Hungry(Kirk)
$\neg$ Hungry(Kirk) $\vee$ Sick(Kirk)
$\therefore$ Sick(Kirk)

3. Sick(Kirk)
$\neg$ Sick(x) $\vee$ Diseased(x)
$\therefore$ Diseased(Kirk)

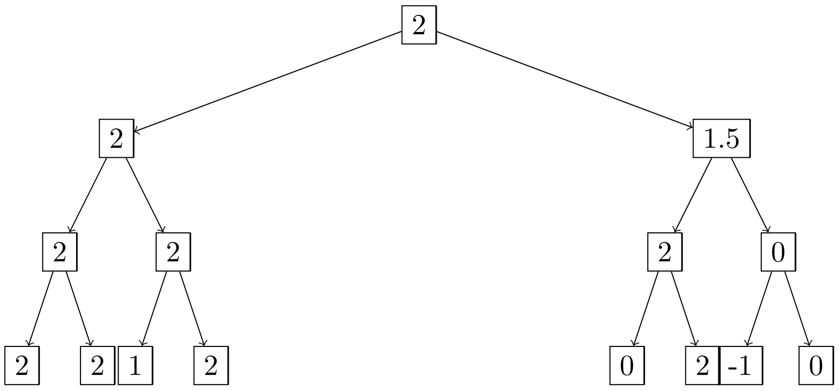
4. Diseased(Kirk)
$\neg$ OnShip(Enterprise, x) $\vee$ $\neg$ Diseased(x)
$\therefore$ $\neg$ OnShip(Enterprise, Kirk)

5. $\neg$ OnShip(Enterprise, Kirk)
$\neg$ Captain(y, x) $\vee$ OnShip(x, y)
$\therefore$ $\neg$ Captain(Enterprise, Kirk)
so we've reached a contradiction.

In words: Kirk ate several whole pizzas for lunch meaning he was hungry. Kirk is only hungry when he is sick. Thus, Kirk is sick, which also means Kirk is diseased. Since the Enterprise doesn't have any diseases, Kirk is not on the Enterprise. Since the caption of a ship must remain on a ship, Kirk cannot be the captain of the Enterprise, a contradiction.

Q5

(a)



The optimal decision at the root is to go left.

(b) Yes, we do need to evaluate the seventh and eighth leaves. Those two leaves have no constraint on them, and could thus take any possible value. For example, if the rightmost leaf node takes value 3 instead of 0, then it changes the root decision to go right.

(c) We only need to evaluate the first four leaves. The last four don't need to be evaluated because going right from the root cannot give a higher expected value than what the root node gets when going left, since the maximum value of a leaf is 2 and that is already attained by going left.

(d) We need to show that the decisions at each level of the search tree are the same.

Let T denote the original search tree and T' denote the tree after scaling the leaf values. We'll show that at every node n' in T' corresponding to n in T, the value of that node v(n') = av(n) + b.

We'll refer to depth in this proof as the distance from the bottom of the tree (the leaves).

Base Case: Clearly true for the leaves since we applied the linear transform to each leaf value.

Now suppose it's true for all nodes up till depth d. We must show it's true for depth d+1.

If the nodes at depth d+1 are chance nodes:

Let α denote the set of actions from a given chance node, n', in state s. Let U'(s,a) denote the expectimax value of taking action r from state s in tree T'.

$$v(n') = \sum_{r\in\alpha} P(r)U'(s,r) = \sum_{r\in\alpha} P(r)(aU(s,r)+b)$$

$$= a\sum_{r\in\alpha} P(r)U(s,r) + b\sum_{r\in\alpha} P(r) = a\sum_{r\in\alpha} P(r)U(s,r) + b = av(n) + b$$

where the second equality follows by the induction hypothesis.

If the nodes at depth d+1 are max nodes:
Expectimax will select the child with the highest expectimax value. Suppose in tree T, this was child c_*. That means the node took on value v(c_*) in T and v(c_*) > v(c_i) for any child $c_i \neq c_*$. By the induction hypothesis, v($c_*^{'}$) = av(c_*) + b > av(c_i) + b = v($c_i^{'}$) for any child $c_i^{'} \neq c_*^{'}$. Thus, the max node will take on value v($c_*^{'}$) = av(c_*) + b and will thus make the same decision in T' as in T.

Thus, by induction, we've shown under a positive linear transformation, expectimax search decisions remain the same.