Nathan Comer Assignment 2
Adversarial Search and First-Order Logic DUE: Tuesday, Dec 6 at 10:00pm (1h55m grace period)

1. (15 pts) Consider the game Connect-4 (https://en.wikipedia.org/wiki/Connect_Four), which is similar to Tic-Tac-Toe. If you were to write a computer program using Alpha-Beta Search with a CUT-OFF test (as detailed in the textbook and Fig 5.7), you would need to create an evaluation function that assessed the current state of the board. This requires a quantification and weighting of features of the current board state (see Sec. 5.4.1). (See clarification at the bottom.)

I will use a combination of functions for my evaluation function.
Define $row(int\ x,\ player\ y)$ to be the number of times that there are x discs in a row on the board for player y such that each of the rows can still be extended (is not blocked), if a row of discs can be extended on both ends it is equivalent to two rows.

$$row(int\ x,\ player\ y) = (\#\ of\ x\ disc\ rows\ extendable\ in\ one\ direction\ for\ player\ y) + 2 * (\#\ of\ x\ disc\ rows\ extendable\ in\ both\ directions\ for\ player\ y)$$

Note: longer rows do not also count as multiple short rows in the definition of $row(int\ x,\ player\ y)$. i.e. a row of 3 discs is not also 2 rows of 2 discs.

Evaluation function: $E_{val}(s) = f_1(s) + f_2(s) + f_3(s) + f_4(s)$

Let: $f_1(s) = row(4,\ self) > 0\ ?\ w_1 : 0$ $w_1 = 100$

$f_2(s) = \sum_{i=2}^{3} row(i,\ self) * (w_2)^i$ $w_2 = 2.5$

$f_3(s) = -\sum_{i=2}^{3} row(i,\ opponent) * (w_3)^i$ $w_3 = 2.0$

$f_4(s) = row(4,\ opponent) > 0\ ?\ -w_4 : 0$ $w_4 = 100$

a. Define that portion of the evaluation function that corresponds to the offense (i.e. reinforce states that are likely to lead to the completion of 4 in a row). Remember to provide a signed weight for each term.
   The first part of the equation ($f_1(s) + f_2(s)$) corresponds to the offense.
   The first function will return w_1 if the player has 4 in a row on this board, meaning this board results in a win for the player, so we will give this function a large weighting so whatever actions were picked that resulted in this board will be chosen.
   The second function sums up the rows the player has on the board. There are two possible row lengths we will account for in this function, two and three discs. Thus this function will return the number of two disc rows given by the definition above, times the weighting to the power of 2, plus the number of three disc rows given by the definition above times the weighting to the power of 3. This provides an exponential factor to the values, being longer rows are exponentially better than shorter rows. To summarize this function it encourages the algorithm to create fewer longer rows instead many shorter rows.
b. Define that portion of the evaluation function that corresponds to the defense (i.e. reinforce states that help to prevent your opponent from completing 4 in a row). Remember to provide a signed weight for each term.
   The second part of the equation ($f_3(s) + f_4(s)$) corresponds to the defense.
   The first function ($f_3(s)$) sums up the rows the opponent has on the board and makes it negative. There are two possible row lengths we will account for in this function, two and three discs. Thus this function will return the number of two disc rows given by the definition above times the weighting to the power of 2, plus the number of three disc rows times the weighting to the power of 3. To summarize this function it encourages the algorithm to try to prevent the opposing player from

getting long rows. Given that longer rows receive exponentially more weighting because they are worth exponentially more than shorter rows.

The second function ($f_4(s)$) will return $-w_4$ if the opposing player has 4 in a row on this board, meaning this board should be prevented no matter what, so we will give this function a large weighting so whatever actions were chosen that resulted in this board will not be performed. This function is under the assumption that it is impossible for both players to have 4 in a row on the same board given that one of the player would have to have already won, or the search would have terminated.

c. Imagine you want to write your program such that it can play at different levels of expertise. How might you modify Search or the Evaluation Function (or both) to produce different levels of play?

-Harder difficulties: I would makes the magnitude of W1 and W4 extremely high so that the algorithm avoids losing at all costs, and does whatever it possibly can to win. I would make the weighting of W2, 2.5, and the weighting of W3, 2. Thus the algorithm would prioritize making it's own row of three over preventing the opponent's row of three, but would prioritize blocking the opponent's row of three over making it's own row of two.
-Easier difficulties: I would decrease the weightings of W2 and W3 so that the algorithm doesn't care as much about longer rows, thus giving the opposing player a better chance of winning. For example if W2 and W3 both equaled 1, longer rows would no longer be weighted more than shorter rows. Thus it may choose to make a row of length two instead of a row of length three. I would also significantly lower the magnitude of W1 and W4 so that the algorithm won't always choose to win even if it can and so it may let the opponent win even if it can be prevented. For example a weighting of 2 for both would prioritize creating/preventing a row of 4 only slightly more than shorter rows.
-intermediary difficulties: Weightings could be interpolated between the values stated for the difficulties above.

2. (9 pts) Express the following in first-order logic using:
Enrolled(student, course) and *EarnedGrade(student, course, grade).*
(Forgot to include the value grade. Consider grade in 0 to 100. Passing grade is anything greater than 55.)

a. Some students took French.

$\exists s,\ s.t.\ Enrolled(s, french)$

b. Every student who takes French passes

Depending upon whether EarnedGrade is a boolean or returns a value it would be:

$\forall s,\ Enrolled(s, french) \Rightarrow EarnedGrade(s, french, grade) > 55$

If EarnedGrade is a boolean representing only that a grade was earned then the logic would instead be:

$\forall s,\ Enrolled(s, french) \Rightarrow EarnedGrade(s, french, grade) \wedge grade > 55$

c. The best score in Greek is always higher than the best score in French.

$\forall EarnedGrade(s1,\ french,\ g1),\ \exists EarnedGrade(s2,\ greek,\ g2)\ s.t.\ g2 > g1$

For every earned grade in french there exists a earned grade in greek such that the grade received in greek is greater than the grade received in french.

3. (12 pts) **Entailment.** Consider the knowledge base: (A --> B) ∧ C

A V B

CNF Form:

(¬A V B)

C

A V B

Decide whether each of the following is true. Briefly justify your answer or Show Your Work.

a. KB ⊨ B

True, B must be true for all models in which the KB is true, (¬A V B) ∧ (A V B) --> B

b. M(KB) ⊆ M(A)

False, A can be true or false when the KB is true. Being we know that B must be true when the KB is true there is no relation between the KB being true and A being true. In (¬A V B), A can be true or false with KB being true and similarly in (A V B), A can be true or false with the KB being true.

c. KB ≡ B ∧ C

True, B must be true if the KB is true because (¬A V B) ∧ (A V B) --> B is true, and C must be true if the KB is true because KB --> C from entry 2 in the KB. Also if B and C are true all entries in the KB are true, thus implying that the KB is true so they are equivalent.

((KB --> B ∧ C) ∧ (B ∧ C --> KB)) ≡ (KB <--> B ∧ C) ≡ (KB ≡ B ∧ C)

d. KB ≡ B

False, The knowledge base implies that B is true but B does not imply that the knowledge base is true. B can be true and if C is false the KB is false.

4. (7 pts) **Forward-Chaining.** Consider the below knowledge base and **use forward-chaining to prove 1 ≤ 6**

1 ≤ 3

3+0 ≤ 4

4 ≤ 6+0

∀x x = x + 0

∀x x + 0 = x

∀xy x = y --> x ≤ y

∀xyz x ≤ y ∧ y ≤ z --> x ≤ z

Starting from 1 working our way to 6

Step 1: $1 \leq 3$

Step 2: $3 = 3 + 0$ **because ∀x x = x + 0**

Step 3: $3 \leq 3 + 0$ **because ∀xy x = y --> x ≤ y** **thus** $1 \leq 3 + 0$ **by ∀xyz x ≤ y ∧ y ≤ z --> x ≤ z**

Step 4: $3 + 0 \leq 4$ **because KB row 2** **thus** $1 \leq 4$ **by ∀xyz x ≤ y ∧ y ≤ z --> x ≤ z**

Step 5: $4 \leq 6 + 0$ **because KB row 3** **thus** $1 \leq 6 + 0$ **by ∀xyz x ≤ y ∧ y ≤ z --> x ≤ z**

Step 6: $6 + 0 = 6$ **because ∀x x + 0 = x**

Step 7: $6 + 0 \leq 6$ **because ∀xy x = y --> x ≤ y** **thus** $1 \leq 6$ **by ∀xyz x ≤ y ∧ y ≤ z --> x ≤ z**

Therefore, $1 \leq 6$

5. (7 pts) Consider the following that represents a few rules of Wumpus World.

S(x,y):Stench at location [x,y]. **W(x,y)**:Wumpus at location [x,y]. **OK(x,y)**:No Wumpus at location [x,y].

KB:

∀xy S(x, y) <--> W(x, y+1) V W(x+1,y) V W(x-1, y)
∀xy ¬W(x,y) <--> OK(x,y)
¬S(2,1)
S(3,1)

a. (5pts) Convert this knowledge base into CNF (Conjunctive Normal Form), which Resolution uses to respond to a query like ASK(KB, OK(1,1))

∀xy S(x, y) <--> W(x, y+1) V W(x+1,y) V W(x-1, y)
(S(x, y) --> W(x, y+1) V W(x+1,y) V W(x-1, y)) ^
(W(x, y+1) V W(x+1,y) V W(x-1, y) --> S(x, y)))
(¬S(x, y) V W(x, y+1) V W(x+1,y) V W(x-1, y)) ^
(¬(W(x, y+1) V W(x+1,y) V W(x-1, y)) V S(x, y)))
(¬S(x, y) V W(x, y+1) V W(x+1,y) V W(x-1, y)) ^
((¬W(x, y+1) ^ ¬W(x+1,y) ^ ¬W(x-1, y)) V S(x, y)))
(¬S(x, y) V W(x, y+1) V W(x+1,y) V W(x-1, y)) ^
(¬W(x, y+1) V S(x, y)) ^ (¬W(x+1,y) V S(x, y)) ^ (¬W(x-1, y) V S(x, y))

∀xy ¬W(x,y) <--> OK(x,y)
¬W(x,y) --> OK(x,y) ^ OK(x,y) --> ¬W(x,y)
(W(x,y) V OK(x,y)) ^ (¬OK(x,y) V ¬W(x,y))

<u>So the new knowledge base in CNF:</u>
∀xy, ¬S(x, y) V W(x, y+1) V W(x+1,y) V W(x-1, y)
∀xy, ¬W(x, y+1) V S(x, y)
∀xy, ¬W(x+1,y) V S(x, y)
∀xy, ¬W(x-1, y) V S(x, y)
∀xy, W(x,y) V OK(x,y)
∀xy, ¬OK(x,y) V ¬W(x,y)
¬S(2,1)
S(3,1)

b. (2pts) If you ASK(KB, OK(2,1)), it will return False. What does that mean?
It means that ¬OK(2,1),
and ¬OK(2,1) --> W(2,1) because ∀xy, W(x,y) V OK(x,y)