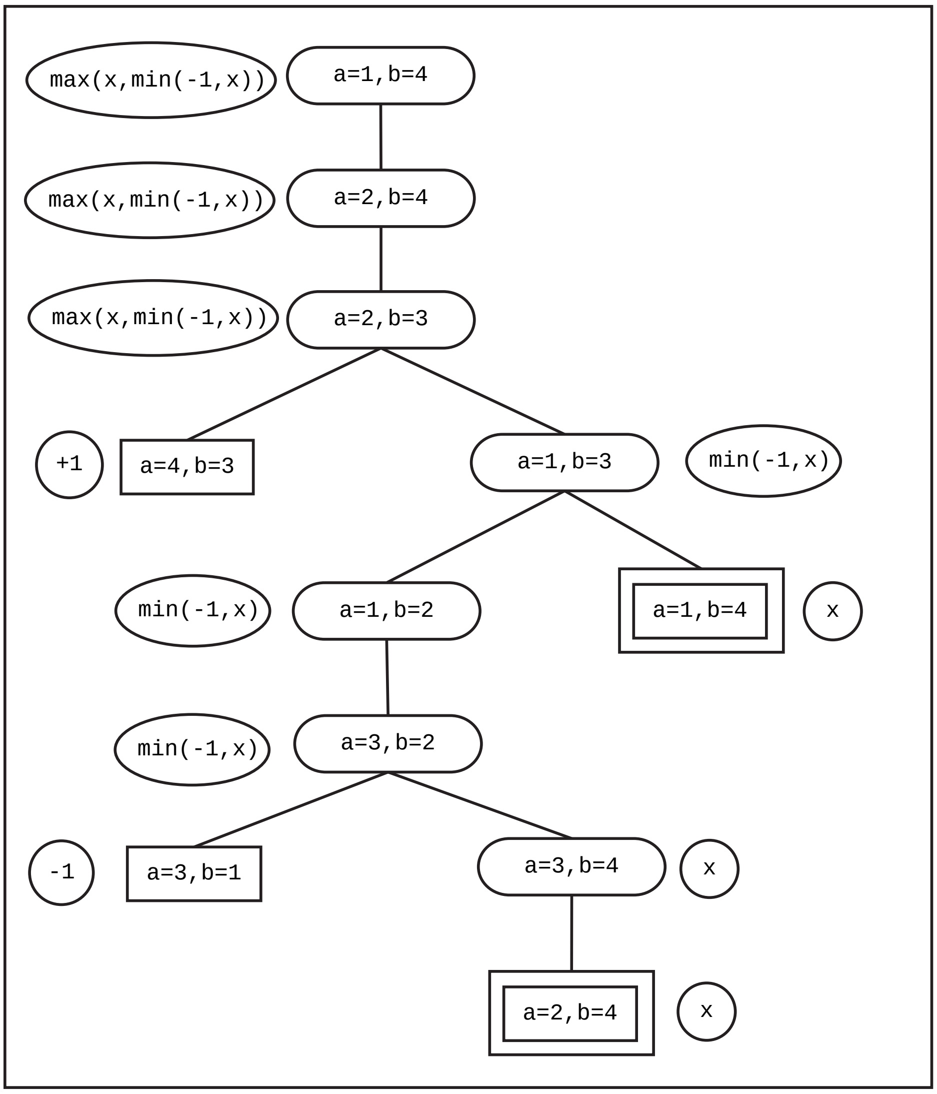
Assignment 2

Maxime CHAMBREUIL
maxime.chambreuil@mail.mcgill.ca

October 6, 2003

Contents

1 Exercise 1: Adversarial search



1.1 Value for loop state

We have assigned the value x to the loop state. Let us now discuss the different value of x :

$$\begin{aligned} x < -1 &\Rightarrow max(1, min(-1, x)) = 1 \Rightarrow \text{ A wins.} \\ -1 \leq x \leq 1 &\Rightarrow max(1, min(-1, x)) = 1 \Rightarrow \text{ A wins.} \\ x > -1 &\Rightarrow max(1, min(-1, x)) = 1 \Rightarrow \text{ A wins.} \end{aligned}$$

So, in this game, A wins whatever the value of x.

1.2 Minimax failure

Thanks to the previous question, we can notice that we can affect any value to x, this does not affect the result of the Minimax algorithm. But we need to affect a value to x if there are loop state in another game.

When we consider a loop state and the reason why a player should choose this state, we can say that any player loses less in choosing to loop instead of losing definitely. That is why I would generally choose x = 0, then each player has the same chance to win.

1.3 n even or odd ?

First, let us demonstrate that the first player who goes over his opponent is winning : Indeed, if you have to go over your opponent, then you will be doing less cases than him to go to your goal case.

If the number of cases is even, A and B will meet at $\frac{n}{2}$ and $\frac{n}{2}+1$ and A has to play. So A goes over B, and A wins.

If the number of cases is odd, A and B will meet at $\frac{n}{2}+1$ and $\frac{n}{2}+2$ and B has to play. So B goes over A, and B wins.

2 Exercise 2: Propositional logic

$$X_{i,j} \text{ is true } \iff \text{[i , j] contains a mine.}$$

2.1 Exactly 2 mines adjacent to $X_{1,1}$

There are exaclty 2 mines adjacent to $X_{1,1}$ if and only if :

$$(X_{1,2} \wedge X_{2,1}) \vee (X_{1,2} \wedge X_{2,2}) \vee (X_{2,1} \wedge X_{2,2}) \iff [X_{1,2} \wedge (X_{2,1} \vee X_{2,2})] \vee (X_{2,1} \wedge X_{2,2})$$

2.2 k mines adjacent to $X_{i,j}$

The method is to enumerate every permutation of k elements in a field of n. Then you suppose that one choice is right, which means that you have the intersection of k propositions. But it can be another choice, so you have to make the union of all choices:

$$\overbrace{\overbrace{(X_{i-1,j-1} \wedge X_{i-1,j} \wedge X_{i-1,j+1} \wedge \cdots)}^{\text{k cases}} \vee \cdots}^{n\times(n-1)\cdots(n-k) \text{ choices}}$$

3 Exercise 3: Predicate logic

3.1 Predicates

- Take (X , Y , Z) : X take Y course in Z.
- Passe (X , Y) : X passes Y course.
- Score (X , Y) : score of X in Y course.
- Student (X) : X is a student.
- French, Greek, Spring 2001 : Constants.

3.2 Some students took French in spring 2001.

∃ X, Student (X) ∧ Take (X , French , Spring 2001)

3.3 Every student who takes French passes it.

∀ X, ∀ Y, Student (X) ∧ Take (X , French , Y) ⇒ Passe (X , French)

3.4 Only one student took Greek in spring 2001.

∃ X, ∀ Y, Student (X) ∧ [Take (X , Greek , Spring 2001) ∨ (X = Y ∧¬ Take (Y , Greek , Spring 2001))]

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

∀ X and Y, ∃ Z and H, Student (X) ∧ Student (Y) ∧ Student (Z) ∧ Student (Z) ∧
Score (X , French) ¡ Score (Z , French) ∧ Score (Y , Greek) ¡ Score (H , Greek) ∧
Score (Z , French) ¡ Score (H , Greek)

3.6 Predicates

- BornIn (X , Y) : X is born in Y.
- Parent (X , Y) : X is a parent of Y.
- Citizen (X , Y , Z) : X is a citizen of Y by Z.
- Resident (X , Y , Z) : X is a resident of Y by Z.
- UK, Birth, Descent : Constants.

3.7 A person born in the UK, each of whose parents is a UK citizen or UK resident, is a UK citizen by birth.

∀ X, Y and Z, BornIn (X , UK) ∧ [Parent (Y , X) ∧ (Citizen (Y , UK , Z) ∨ Resident (Y , UK , Z))]
⇒ Citizen (X , UK , Birth)

3.8 A person born outside the UK, each of whose parents is a UK citizen by birth, is a UK citizen by descent.

∀ X, ∃ Y, ¬ BornIn (X , UK) ∧ [Parent (Y , X) ∧ (Citizen (Y , UK , Birth)]
⇒ Citizen (X , UK , Descent)

4 Exercise 4: Programming question - Knockabout

After playing a few times (I have never heard about and played this game), my heuristic is the distance between the opponent die and the gutter. To be clear enough, here are the basic steps of my algorithm :

- I look for all opponent die and keep the furthest from the gutter
- I look for all my die, and keep the closest to the opponent die
- I move my die in the direction of the opponent die