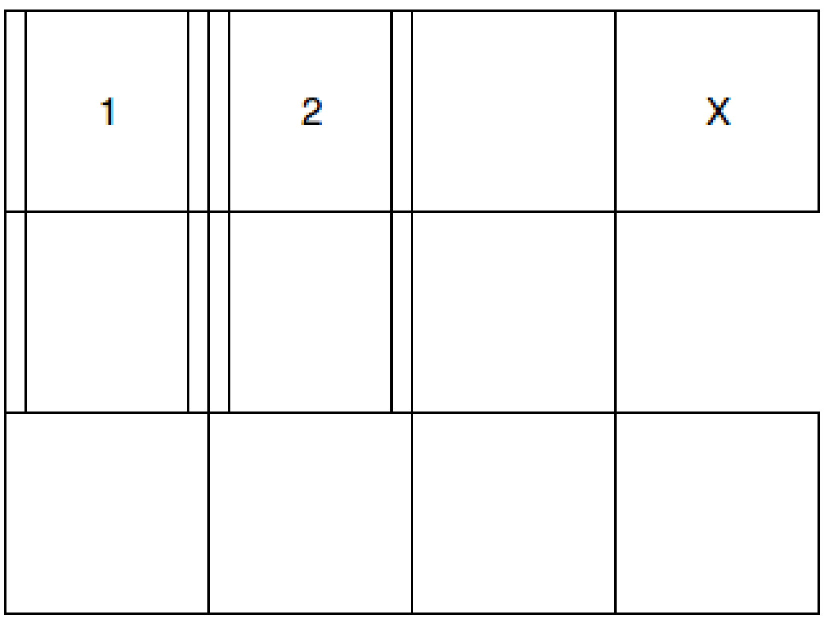
# COMP 424 – Tutorial 2

## Practice Questions for Assignment 2

~ Nicolas Angelard-Gontier ~
nicolas.angelard-gontier@mail.mcgill.ca

# 1. Searching Under Uncertainty

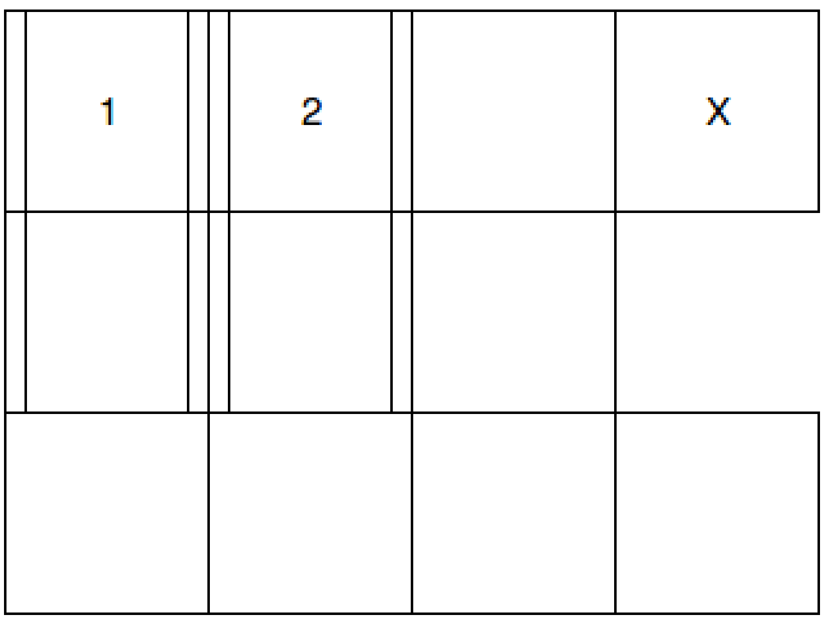
You and your friend are trapped in a maze described below.




(a) You have **no idea** where you are. What is the total number of beliefs in this domain?

# 1. Searching Under Uncertainty

You and your friend are trapped in a maze described below.




(a) You have **no idea** where you are. What is the total number of beliefs in this domain?

<u>Answer</u>:

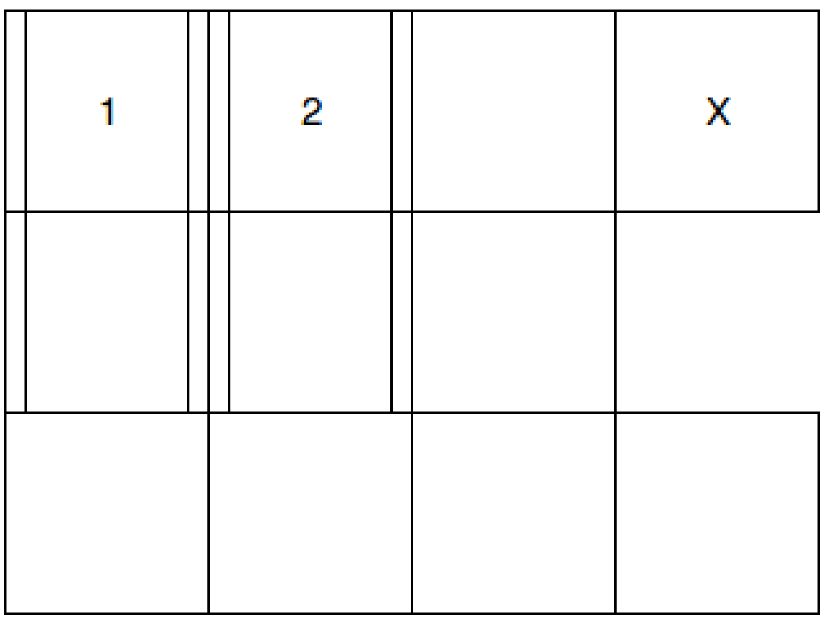
Your belief space is 2 to the power (number of cells) – 1

- power 2 bcs each cell is a binary variable: you are here or you are not
- -1 because there is no empty state (i.e.: you HAVE TO be somewhere)

$2^{11} - 1 = 2{,}047$

# 1. Searching Under Uncertainty

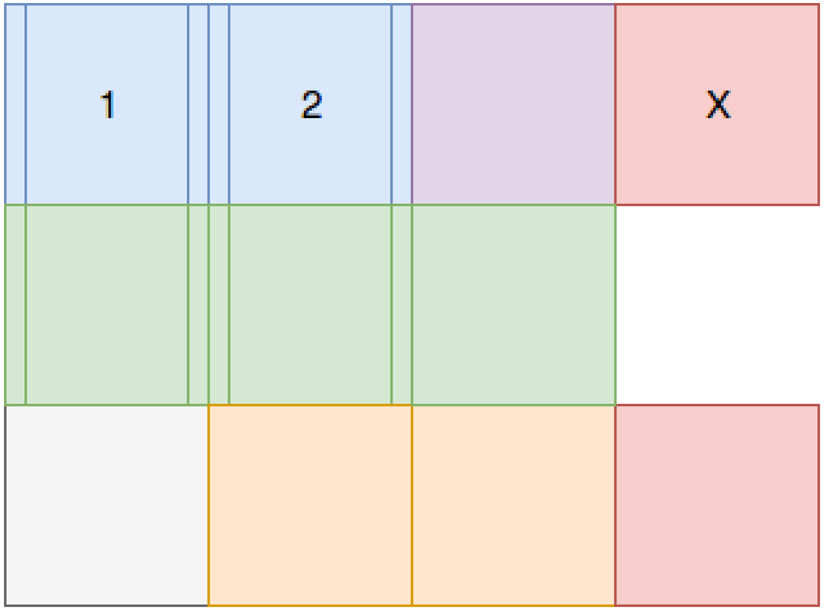
You and your friend are trapped in a maze described below.




(b) You can only perceive the neighboring walls. How many distinct percepts are possible in this domain? What are the unique percept  that indicate exactly where you are?

# 1. Searching Under Uncertainty

You and your friend are trapped in a maze described below.




(b) You can only perceive the neighboring walls. How many distinct percepts are possible in this domain? What are the unique percept  that indicate exactly where you are?

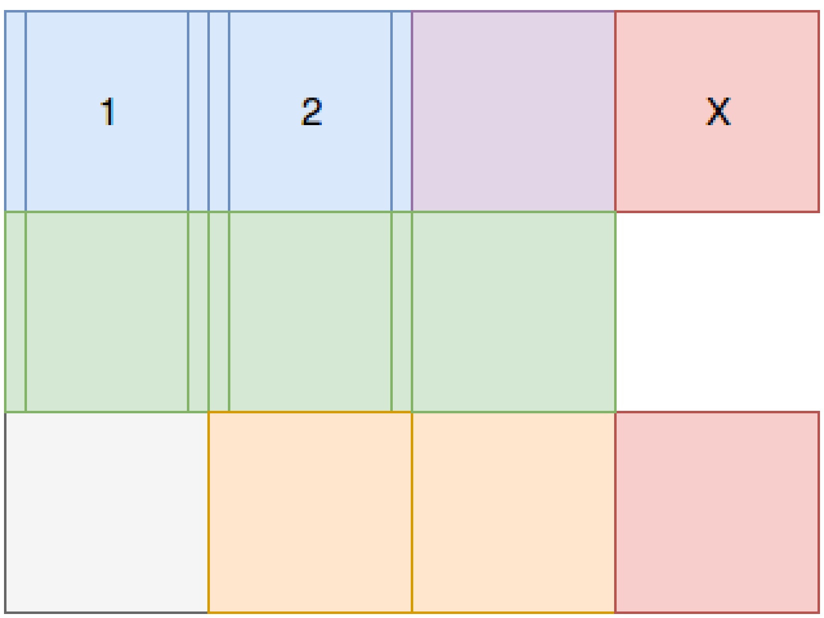
<u>Answer</u>:

6 distinct percepts

- cells 1 & 2 – walls : LEFT, UP, RIGHT
- the cell to the right of 2 – walls : LEFT, UP –> <u>unique percept</u>
- the three cells bellow – walls : LEFT, RIGHT
- the bottom-right cell – walls: LEFT, BOTTOM –> <u>unique percept</u>
- the two cells to the right – walls: BOTTOM
- the two cells at the extreme right – walls: BOTTOM, RIGHT, UP

# 1. Searching Under Uncertainty

You and your friend are trapped in a maze described below.

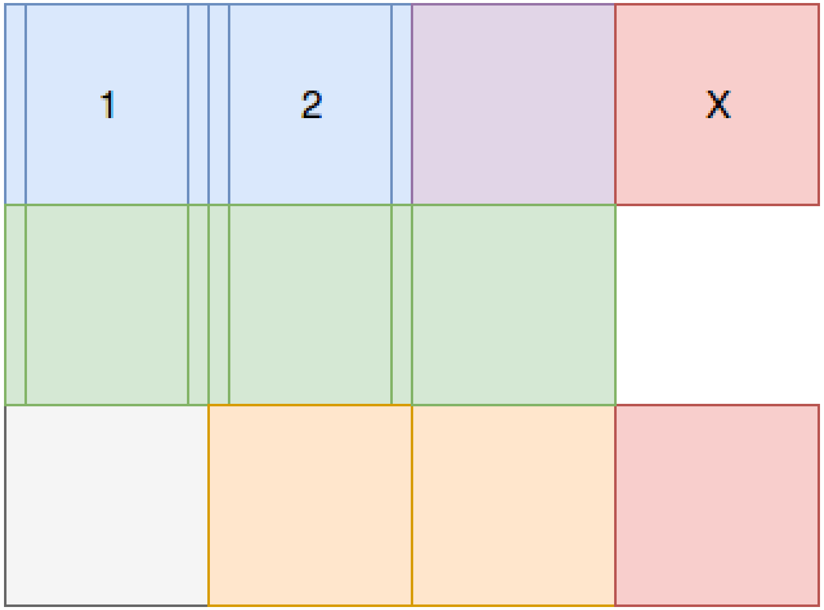



(c) You are either at position 1 or 2. You must get out at the exit X. Your friend is not with you, he is at the other position. You can take 4 actions at each state: UP, DOWN, LEFT, RIGHT. If you hit a wall, nothing happens.

Find the **fastest** conformant plan that can take you out for sure.

# 1. Searching Under Uncertainty

You and your friend are trapped in a maze described below.




(c) You are either at position 1 or 2. You must get out at the exit X. Your friend is not with you, he is at the other position. You can take 4 actions at each state: UP, DOWN, LEFT, RIGHT. If you hit a wall, nothing happens.

Find the **fastest** conformant plan that can take you out for sure.

<u>Answer</u>:

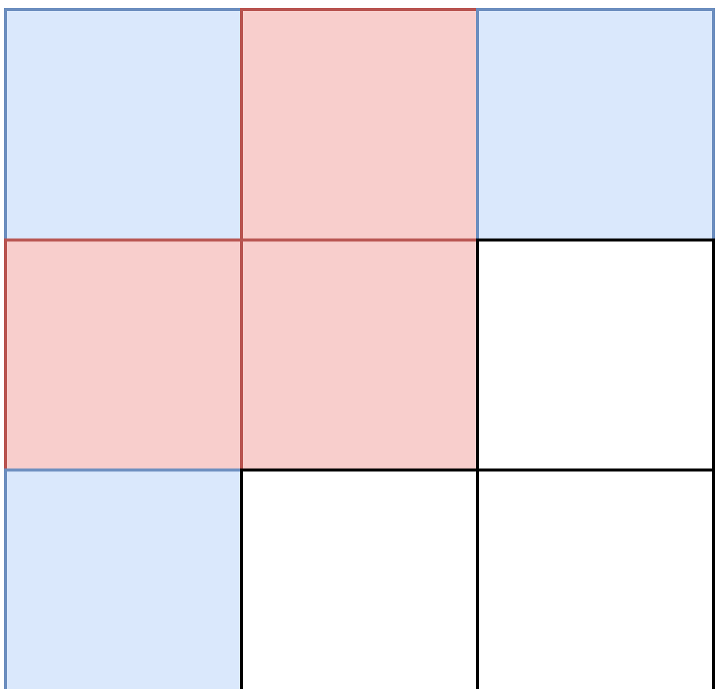
DOWN – DOWN – LEFT –

(now you know exactly where you are!)

- RIGHT – RIGHT – UP – UP – RIGHT

# 2. Game playing

Modified Tic-Tac-Toe game:
<u>You win if any row or column is filled by the same color</u>. You lose in other cases.
You place BLUE boxes.
Start state is:

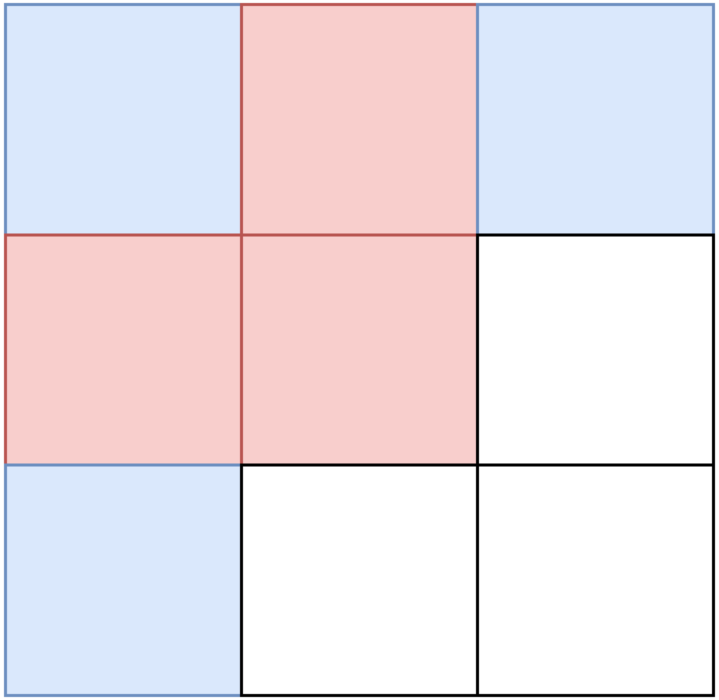


You are the next player to play.

(a) Draw the game tree from this state. Ignore symmetric states.

# 2. Game playing

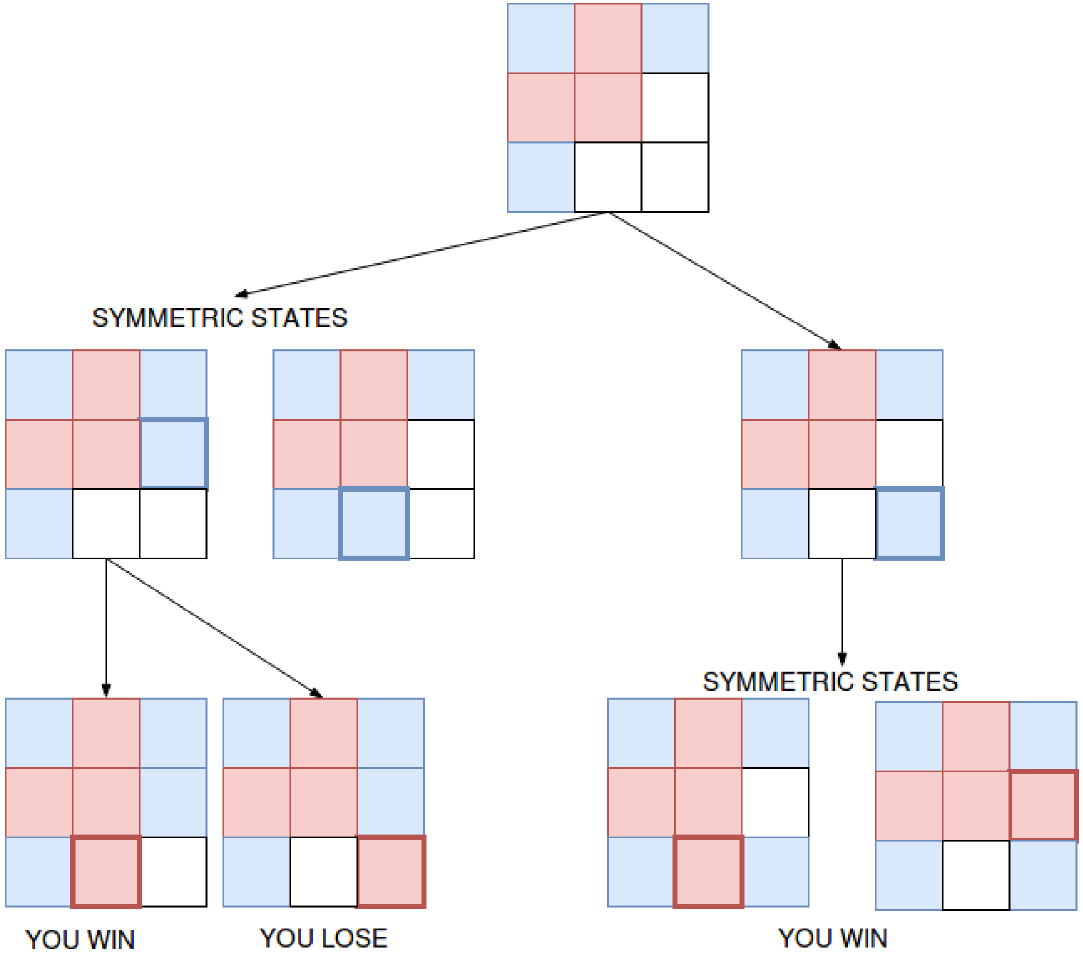
Modified Tic-Tac-Toe game:
<u>You win if any row or column is filled by the same color</u>. You lose in other cases.
You place BLUE boxes.
Start state is:



You are the next player to play.

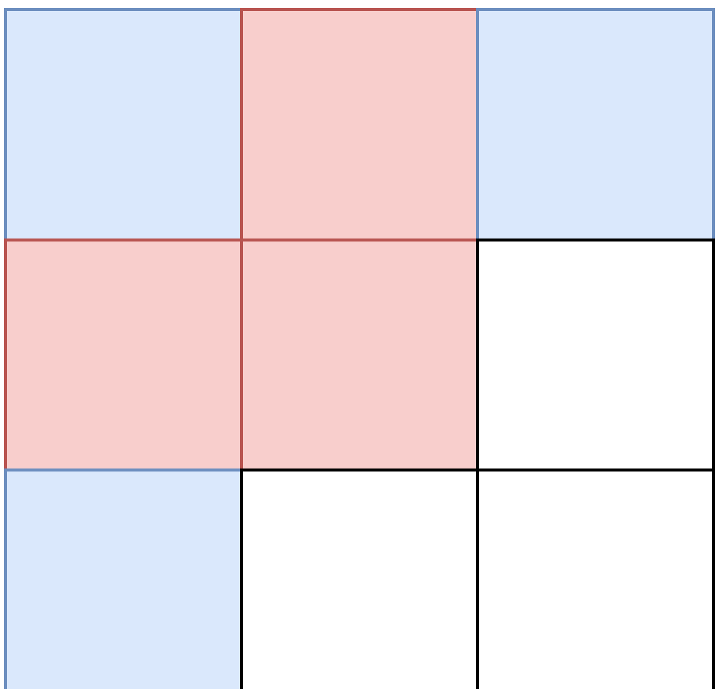
(a) Draw the game tree from this state. Ignore symmetric states.

<u>Answer</u>:



# 2. Game playing

Modified Tic-Tac-Toe game:
<u>You win if any row or column is filled by the same color</u>. You lose in other cases.
You place BLUE boxes.
Start state is:

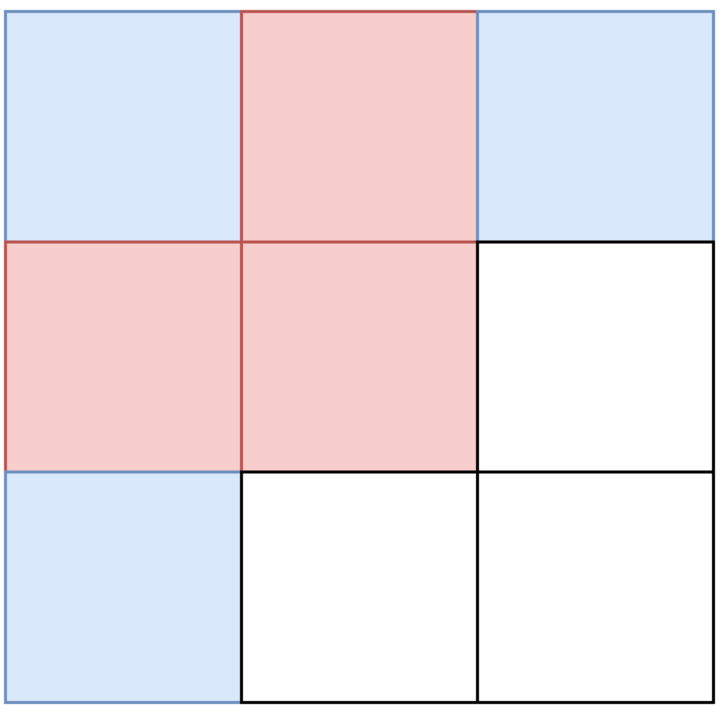


You are the next player to play.

(b) Apply minimax algorithm. How would the game play out?

# 2. Game playing

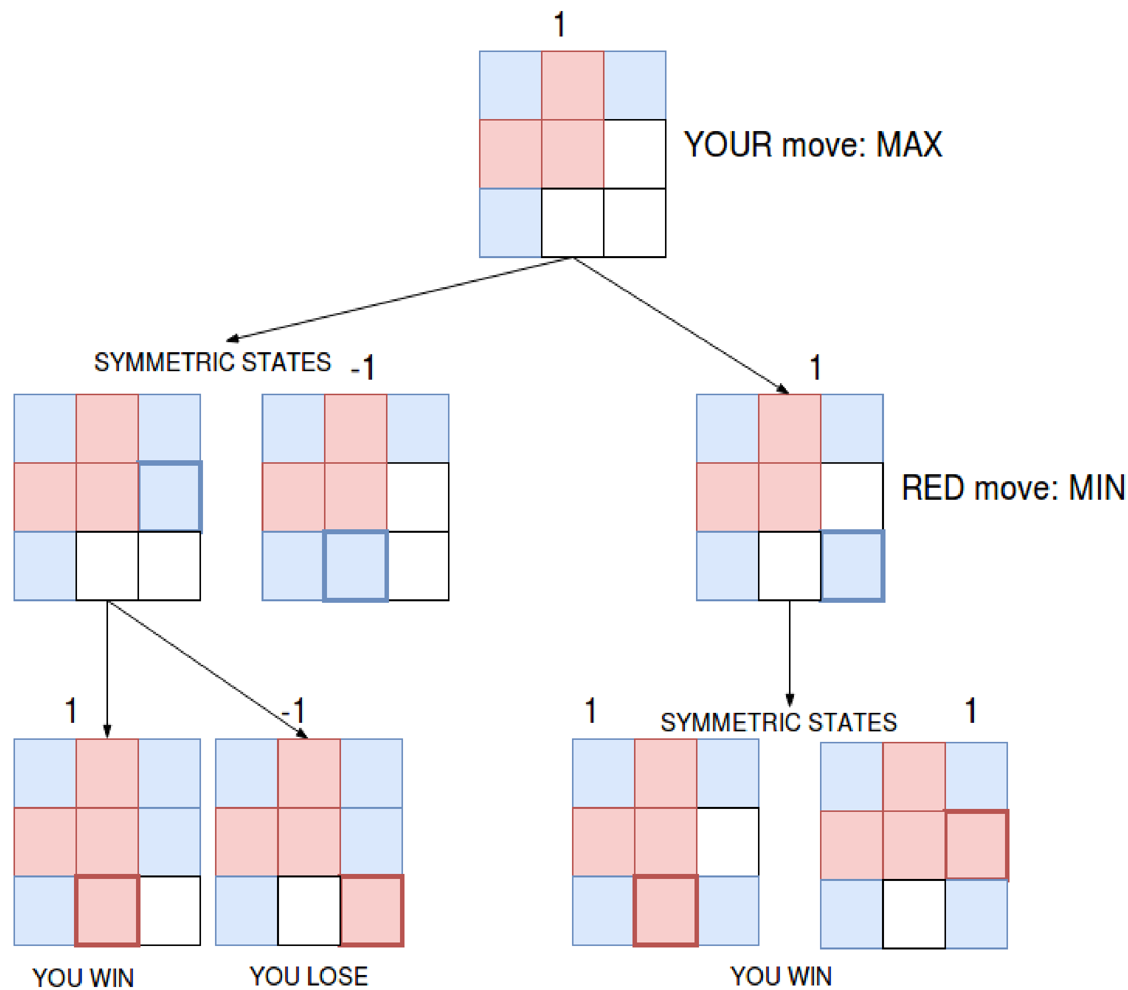
Modified Tic-Tac-Toe game:
<u>You win if any row or column is filled by the same color</u>. You lose in other cases.
You place BLUE boxes.
Start state is:



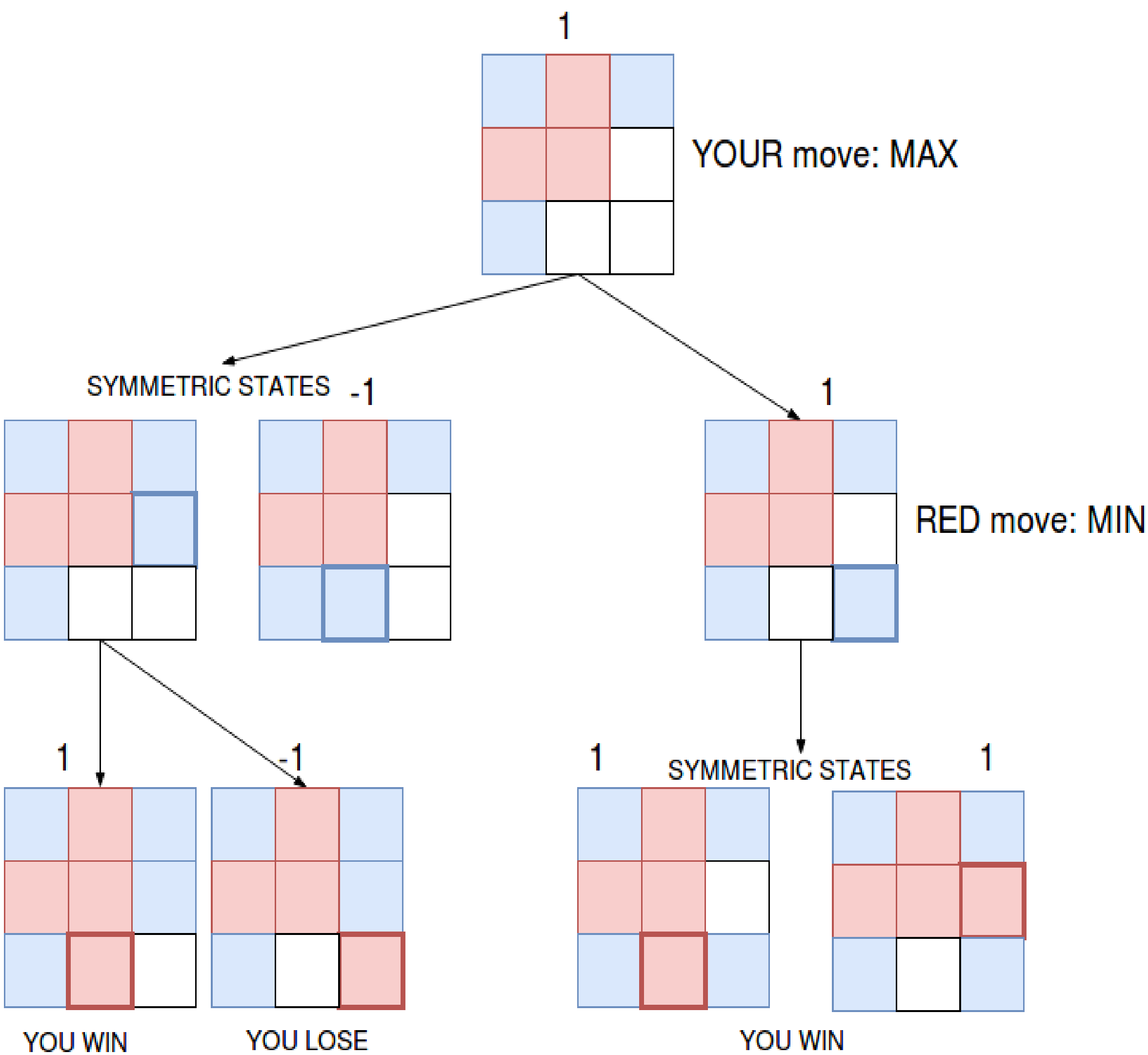
You are the next player to play.

(b) Apply minimax algorithm. How would the game play out?

<u>Answer</u>: You win.

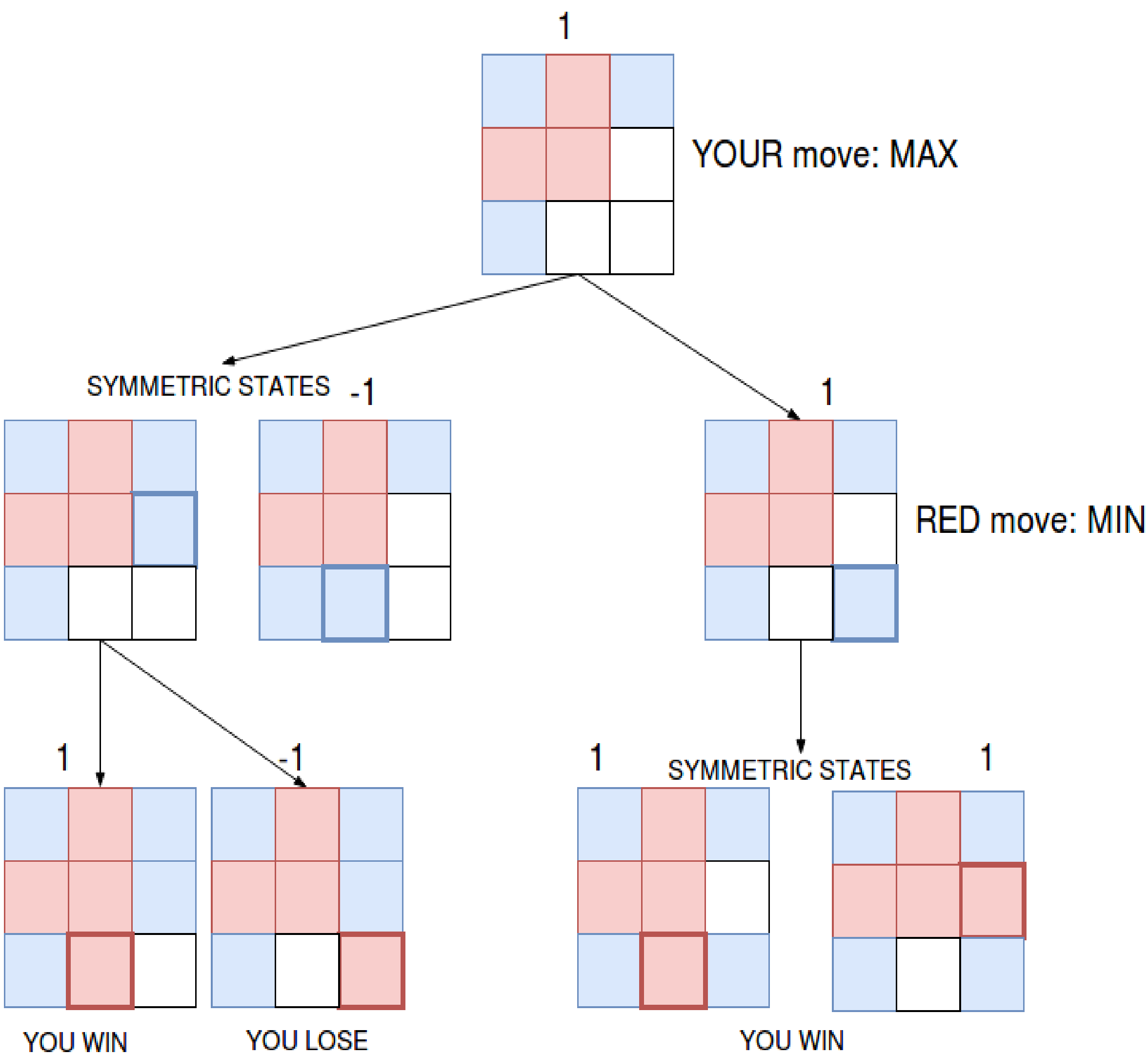


# 2. Game playing




(c) how many game state can be removed if using alpha-beta pruning in the <u>most optimal case</u>?

# 2. Game playing




(c) how many game state can be removed if using alpha-beta pruning in the <u>most optimal case</u>?

<u>Answer</u>:
My move: I chose the right branch, I see that I can win. I can skip the neighboring nodes/sub-tree.
In this case, there are **3 states** that can be prunned.

# 3. Propositional Logic

(a) How many models are there that satisfy each of the following?

- A or B
- A and B
- A => B

# 3. Propositional Logic

(a) How many models are there that satisfy each of the following?

- A or B
- A and B
- A => B

<u>Answer</u>:

A model is a possible arrangement so that the whole thing is true

- A or B
  either A is true, or B is true, or both are true: **3**
- A and B
  only when A is true and B is true: **1**
- A => B
  same as (not A) or B, so either B is true, or A is false, or both: **3**

| A | B | A or B |
|---|---|---|
| T | T | T |
| T | F | T |
| F | T | T |
| F | F | F |

| A | B | A and B |
|---|---|---|
| T | T | T |
| T | F | F |
| F | T | F |
| F | F | F |

| A | B | A => B |
|---|---|---|
| T | T | T |
| T | F | F |
| F | T | T |
| F | F | T |

# 3. Propositional Logic

(b) Which of the following is Valid, Unsatisfiable, or Satisfiable?

- (A and B) or (not A or not B)

- (A and B) or (not A and not B)

- A and (A => not A)

# 3. Propositional Logic

(b) Which of the following is Valid, Unsatisfiable, or Satisfiable?

- (A and B) or (not A or not B)
  Valid
- (A and B) or (not A and not B)
  Satisfiable
- A and (A => not A)
  Unsatisfiable

| A | B | A and B | not A or not B | sentence |
|---|---|---|---|---|
| T | T | T | F | T |
| T | F | F | T | T |
| F | T | F | T | T |
| F | F | F | T | T |

| A | B | A and B | not A and not B | sentence |
|---|---|---|---|---|
| T | T | T | F | T |
| T | F | F | F | F |
| F | T | F | F | F |
| F | F | F | T | T |

| A | B | not A | A => not A | sentence |
|---|---|---|---|---|
| T | T | F | F | F |
| T | F | F | F | F |
| F | T | T | T | F |
| F | F | T | T | F |

# 3. Propositional Logic

NOTE: in the assignment you have **entailment** : A |= B

=> “B is true in all cases where A is true.”

Examples:  valid / satisfiable / unsatisfiable ?

False |= B
False is always false anyway, so:

True |= B
True is always true so:

# 3. Propositional Logic

NOTE: in the assignment you have **entailment** : A |= B

=> "B is true in all cases where A is true."  ~ not A or (A and B)

Examples:  valid / satisfiable / unsatisfiable ?

False |= B
False is always false anyway, so: valid

True |= B
True is always true so: check that A is also always true, otherwise its unsatisfiable

# 4. First Order Logic

Alice and Bob are hungry. They are eating Sushi or Pasta.
Alice eats the opposite of Bob
Bob eats Sushi

Use the function Eat(x, y) to indicate that “x eats y”

(a) Identify the set of constants and variables. Then define the relevant predicates and translate the facts listed above into first order logic.

# 4. First Order Logic

Alice and Bob are hungry. They are eating Sushi or Pasta.
Alice eats the opposite of Bob
Bob eats Sushi

Use the function Eat(x, y) to indicate that “x eats y”

(a) Identify the set of constants and variables. Then define the relevant predicates and translate the facts listed above into first order logic.

<u>Answer</u>:
Constant: X = {Alice, Bob}    Y = {Sushi, Pasta}
Variable:   person x,              food type y

1. $\forall x \in X \; Eat(x, Sushi) \vee Eat(x, Pasta)$
2. $\forall y \in Y \; Eat(Bob, y) \Leftrightarrow \neg Eat(Alice, y)$
3. $Eat(Bob, Sushi)$

# 4. First Order Logic

Alice and Bob are hungry.
They are eating Sushi or Pasta.
Alice eats the opposite of Bob
Bob eats Sushi

$$\forall x \in X\ Eat(x, Sushi) \vee Eat(x, Pasta)$$
$$\forall\ y \in Y\ Eat(Bob, y) \Leftrightarrow \neg Eat(Alice, y)$$
$$Eat(Bob, Sushi)$$

Constant: X = {Alice, Bob}    Y = {Sushi, Pasta}
Variable: person x,    food type y

(b) Convert all facts from part (a) to CNF. Number each clause.

# 4. First Order Logic

Alice and Bob are hungry.
They are eating Sushi or Pasta.
Alice eats the opposite of Bob
Bob eats Sushi

$$\forall x \in X \; Eat(x, Sushi) \vee Eat(x, Pasta)$$
$$\forall y \in Y \; Eat(Bob, y) \Leftrightarrow \neg Eat(Alice, y)$$
$$Eat(Bob, Sushi)$$

Constant: X = {Alice, Bob}    Y = {Sushi, Pasta}
Variable: person x,    food type y

(b) Convert all facts from part (a) to CNF. Number each clause.

<u>Answer</u>:
The Knowledge Base (KB) is made of:

1. $Eat(x, Sushi) \vee Eat(x, Pasta)$
2. $\neg Eat(Bob, y) \vee \neg Eat(Alice, y)$
3. $Eat(Alice, y) \vee Eat(Bob, y)$
4. $Eat(Bob, Sushi)$

# 4. First Order Logic

1. $Eat(x, Sushi) \vee Eat(x, Pasta)$
2. $\neg Eat(Bob, y) \vee \neg Eat(Alice, y)$
3. $Eat(Alice, y) \vee Eat(Bob, y)$
4. $Eat(Bob, Sushi)$

(c) Query: Is Alice eating Pasta? Answer the query by using proof by resolution.

# 4. First Order Logic

1. $Eat(x, Sushi) \vee Eat(x, Pasta)$
2. $\neg Eat(Bob, y) \vee \neg Eat(Alice, y)$
3. $Eat(Alice, y) \vee Eat(Bob, y)$
4. $Eat(Bob, Sushi)$

(c) Query: Is Alice eating Pasta? Answer the query by using proof by resolution.

<u>Answer</u>:

query: $\alpha = Eat(Alice, Pasta)$ $\neg\alpha = \neg Eat(Alice, Pasta)$

let's show that $KB \wedge \neg\alpha$ is unsatisfiable:

(i) use [4. and 2.] with $\sigma = \{y/Sushi\}$ : $\neg Eat(Alice, Sushi)$

(ii) use [(i) and 1.] with $\sigma = \{x, Alice\}$ : $Eat(Alice, Pasta)$

$KB \wedge \neg\alpha = Eat(Alice, Pasta) \wedge \neg Eat(Alice, Pasta)$

This is unsatisfiable therefore Query is True.

Questions ?

~ Nicolas Angelard-Gontier ~
nicolas.angelard-gontier@mail.mcgill.ca