

# 1 Discussion Based Warm-ups

(a) Recall the definition of satisfiability and entailment.

- **Satisfiability:**

- **Entailment:**

(b) What is the difference between satisfiability and entailment? Can you use one to prove the other?

## 2 SATurdays are for everyone

1. Determine whether the sentences below are satisfiable or unsatisfiable (using any method you like).

(a)  $(\neg(y \vee \neg y) \vee x) \wedge (x \vee (z \iff \neg z))$

(b)  $\neg(x \vee \neg(x \wedge (z \vee \top))) \implies \neg(y \wedge (\neg y \vee (\top \implies \perp)))$

(c)  $((\top \iff \neg(x \vee \neg x)) \vee z) \vee z \wedge \neg(z \wedge ((z \wedge \neg z) \implies x))$

2. Suppose  $A \models B$ . Which of the following statements must be true for all truth assignments to  $A$  and  $B$ ?

(a)  $A \wedge B$

(c)  $B \Rightarrow A$

(e)  $B$

(b)  $A \Rightarrow B$

(d)  $A \vee B$

3. How would we formulate the SAT problem as a CSP? What are the variables? Domains? Constraints?

4. Suppose we have an algorithm which determines whether a sentence is satisfiable or not. Given two sentences  $A$  and  $B$ , how could we determine whether  $A \models B$ ?

5. Determine whether the sentence below is satisfiable or unsatisfiable using DPLL. Break ties by assigning variables in alphabetical order, starting with false. If satisfiable, what model does the algorithm find?

$$(A \vee B) \wedge (B \vee C \vee D) \wedge (\neg A \vee \neg B \vee C) \wedge (\neg A \vee \neg C \vee \neg D) \wedge A \wedge (C \vee \neg D)$$

### 3 Wandering in Wumpus World

We bring together what we have learned in lecture as well as the ideas of search so far in order to construct wumpus world agents that use propositional logic. The first step is to enable the agent to deduce, to the extent possible, the state of the world given its percept history. This requires writing down a complete logical model of the effects of actions. We also show how the agent can keep track of the world efficiently without going into the percept history for each inference. Finally, we show how the agent can use logical inference to construct plans that are guaranteed to achieve its goals.

Try it out: <http://thiagodnf.github.io/wumpus-world-simulator/>

Throughout this question, we will present several screenshots from the Wumpus World simulator linked previously. In each of these, assume that you *do* have an arrow on hand (as an extra exercise, consider how the answers might be different if you did not have an arrow). Also, note that the location of the explorer can be ignored. We just tried to place him somewhere where he wouldn't be blocking the text!

Refer to the code below to answer the questions:

```

function HYBRID-WUMPUS-AGENT(percept) returns an action
  inputs: percept, a list, [stench, breeze, glitter, bump, scream]
  persistent: KB, a knowledge base, initially the atemporal "wumpus physics"
               t, a counter, initially 0, indicating time
               plan, an action sequence, initially empty

  TELL(KB, MAKE-PERCEPT-SENTENCE(percept, t))
  TELL the KB the temporal "physics" sentences for time t
  safe  $\leftarrow \{[x, y] : \text{ASK}(\text{KB}, \text{OK}_{x,y}^t) = \text{true}\}$ 
  if ASK(KB, Glittert) = true then
    plan  $\leftarrow [\text{Grab}] + \text{PLAN-ROUTE}(\text{current}, \{[1,1]\}, \text{safe}) + [\text{Climb}]$ 
  if plan is empty then
    unvisited  $\leftarrow \{[x, y] : \text{ASK}(\text{KB}, L_{x,y}^{t'}) = \text{false} \text{ for all } t' \leq t\}$ 
    plan  $\leftarrow \text{PLAN-ROUTE}(\text{current}, \text{unvisited} \cap \text{safe}, \text{safe})$ 
  if plan is empty and ASK(KB, HaveArrowt) = true then
    possible_wumpus  $\leftarrow \{[x, y] : \text{ASK}(\text{KB}, \neg W_{x,y}) = \text{false}\}$ 
    plan  $\leftarrow \text{PLAN-SHOT}(\text{current}, \text{possible\_wumpus}, \text{safe})$ 
  if plan is empty then // no choice but to take a risk
    not_unsafe  $\leftarrow \{[x, y] : \text{ASK}(\text{KB}, \neg \text{OK}_{x,y}^t) = \text{false}\}$ 
    plan  $\leftarrow \text{PLAN-ROUTE}(\text{current}, \text{unvisited} \cap \text{not\_unsafe}, \text{safe})$ 
  if plan is empty then
    plan  $\leftarrow \text{PLAN-ROUTE}(\text{current}, \{[1, 1]\}, \text{safe}) + [\text{Climb}]$ 
  action  $\leftarrow \text{POP}(\text{plan})$ 
  TELL(KB, MAKE-ACTION-SENTENCE(action, t))
  t  $\leftarrow t + 1$ 
  return action

function PLAN-ROUTE(current, goals, allowed) returns an action sequence
  inputs: current, the agent's current position
           goals, a set of squares; try to plan a route to one of them
           allowed, a set of squares that can form part of the route

  problem  $\leftarrow \text{ROUTE-PROBLEM}(\text{current}, \text{goals}, \text{allowed})$ 
  return A*-GRAPH-SEARCH(problem)

```

Figure 1: Hybrid-Wumpus-Agent from AIMIA 3rd ed. It uses a propositional knowledge base to infer the state of the world, and a combination of problem-solving search and domain-specific code to decide what actions to take.

(a) Consider the following Wumpus World state:

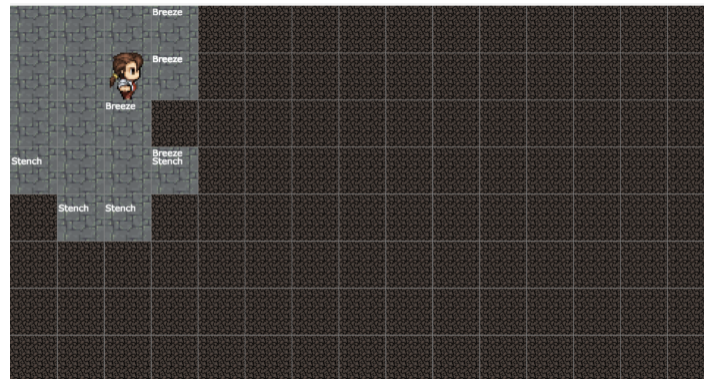


Figure 2: Entailment vs. Satisfiability?

Based on our previous discussion around entailment and satisfiability, identify locations where our knowledge base entails that there must be a Wumpus, Pit, or safe path. Additionally, identify locations where Wumpuses, Pit, and safe paths are not entailed but could be satisfied.

- (b) Take a moment to familiarize yourself with the pseudocode below to understand how we might decide to act in Wumpus World. You'll notice that we have labeled the key decision-making portions of this code, and that different decisions need to be made given the state of our knowledge base.

On the next page, match each of the following states to one of the labeled code chunks in the pseudocode, and explain your reasoning.

```

function HYBRID-WUMPUS-AGENT(percept) returns an action
inputs: percept, a list, [stench,breeze,glitter,bump,scream]
persistent: KB, a knowledge base, initially the atemporal "wumpus physics"
               t, a counter, initially 0, indicating time
               plan, an action sequence, initially empty

TELL(KB, MAKE-PERCEPT-SENTENCE(percept, t))
TELL the KB the temporal "physics" sentences for time t
safe  $\leftarrow \{[x, y] : \text{ASK}(\text{KB}, \text{OK}_{x,y}^t) = \text{true}\}$ 
if ASK(KB, Glittert) = true then
    plan  $\leftarrow$  [Grab] + PLAN-ROUTE(current, {[1,1]}, safe) + [Climb]
if plan is empty then
    unvisited  $\leftarrow \{[x, y] : \text{ASK}(\text{KB}, L_{x,y}^{t'}) = \text{false} \text{ for all } t' \leq t\}$ 
    plan  $\leftarrow$  PLAN-ROUTE(current, unvisited  $\cap$  safe, safe)
if plan is empty and ASK(KB, HaveArrowt) = true then
    possible_wumpus  $\leftarrow \{[x, y] : \text{ASK}(\text{KB}, \neg W_{x,y}) = \text{false}\}$ 
    plan  $\leftarrow$  PLAN-SHOT(current, possible_wumpus, safe)
if plan is empty then // no choice but to take a risk
    not_unsafe  $\leftarrow \{[x, y] : \text{ASK}(\text{KB}, \neg \text{OK}_{x,y}^t) = \text{false}\}$ 
    plan  $\leftarrow$  PLAN-ROUTE(current, unvisited  $\cap$  not_unsafe, safe)
if plan is empty then
    plan  $\leftarrow$  PLAN-ROUTE(current, {[1, 1]}, safe) + [Climb]
    action  $\leftarrow$  POP(plan)
TELL(KB, MAKE-ACTION-SENTENCE(action, t))
t  $\leftarrow$  t + 1
return action

function PLAN-ROUTE(current, goals, allowed) returns an action sequence
inputs: current, the agent's current position
          goals, a set of squares; try to plan a route to one of them
          allowed, a set of squares that can form part of the route

problem  $\leftarrow$  ROUTE-PROBLEM(current, goals, allowed)
return A*-GRAPH-SEARCH(problem)

```

Figure 3: Hybrid-Wumpus-Agent from AIMIA 3rd ed. It uses a propositional knowledge base to infer the state of the world, and a combination of problem-solving search and domain-specific code to decide what actions to take.

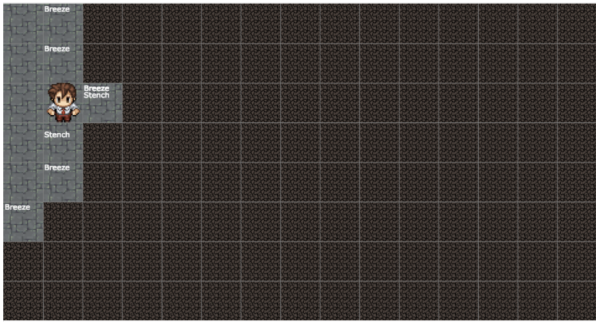
State	Code Chunk
	
	
	
	

Table 1: Which code chunk is applicable for each of these states?

## 4 Journey to Success(or-State Axioms)

(a) First, let's review some definitions. What are successor-state axioms?

(b) Consider the following Mini Pacman grid. In this simplified world, the only available actions are *Left*, *Right*, and *Stay*. The only possible states are  $Pacman_{(1,1)}$  and  $Pacman_{(2,1)}$ . If Pacman tries to move into a wall, he will stay in the same state.

Notice that Pacman's state and actions are both fluent, so we can set up successor-state axioms to define how Pacman moves in this world. Write the successor-state axiom corresponding to Figure 4.

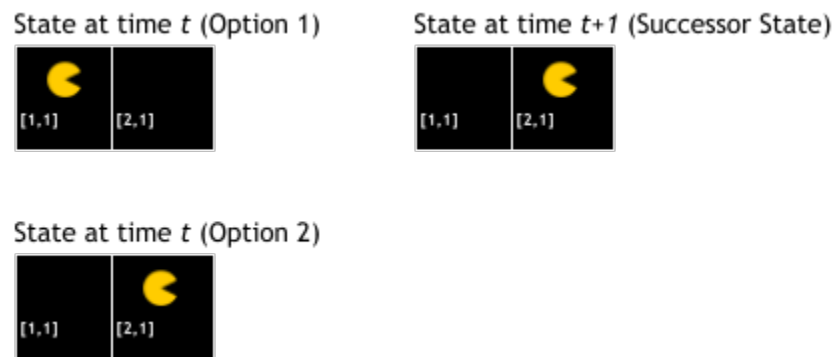


Figure 4: Mini Pacman Grid