1 Conceptual Review

(a) Vocabulary check: Are you familiar with the following terms?

- Symbols:
- Operators:
- Sentences:
- Equivalence:
- Literals:
- Knowledge Base:
- Entailment:
- Clauses (Definite vs. Horn Clauses):
- Model Checking:
- Theorem Proving:
- Modus Ponens:

(b) Recall the definitions of satisfiability and entailment.

- **Satisfiability**:

- **Entailment**:

(c) What is the difference between satisfiability and entailment?

(d) Suppose $A \models B$. Consider all models assigning values to variables in sentences A and B. Which of the following sentences must be true in all possible models (even if either or both A/B are false)?

(a) $A \wedge B$
(b) $A \Rightarrow B$
(c) $B \Rightarrow A$
(d) $A \vee B$
(e) B

(e) Determine which of the following are correct, and explain your reasoning.

- $(A \vee B) \models (A \Rightarrow B)$

- $A \iff B \models A \vee \neg B$

- $(A \vee B) \wedge \neg(A \Rightarrow B)$ is satisfiable

(f) How would we formulate the SAT problem as a CSP? What are the variables? Domains? Constraints?

(g) 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 $A \models B$?

2 SATurdays are for everyone

- P
- $\neg P \vee Q$
- $P \vee \neg Q \vee R$
- $\neg Q \vee R$

Prove R to be true using resolution.

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/ Note that there are some slight differences between this online version and the version we describe below.

Throughout this question, we will present several screenshots from the Wumpus World simulator linked previously. Note that the location of the explorer can be ignored. We just tried to place him somewhere off screen!

Recall that an agent in the Wumpus World has access to the following percepts:

- In the square containing the wumpus and in the directly (not diagonally) adjacent squares, the agent will perceive a Stench.
- In the squares directly adjacent to a pit, the agent will perceive a Breeze.
- In the square where the gold is, the agent will perceive a Glitter.
- When an agent walks into a wall, it will perceive a Bump.
- When the wumpus is killed, it emits a woeful Scream that can be perceived anywhere in the cave.

(a) Consider the following Wumpus World state:

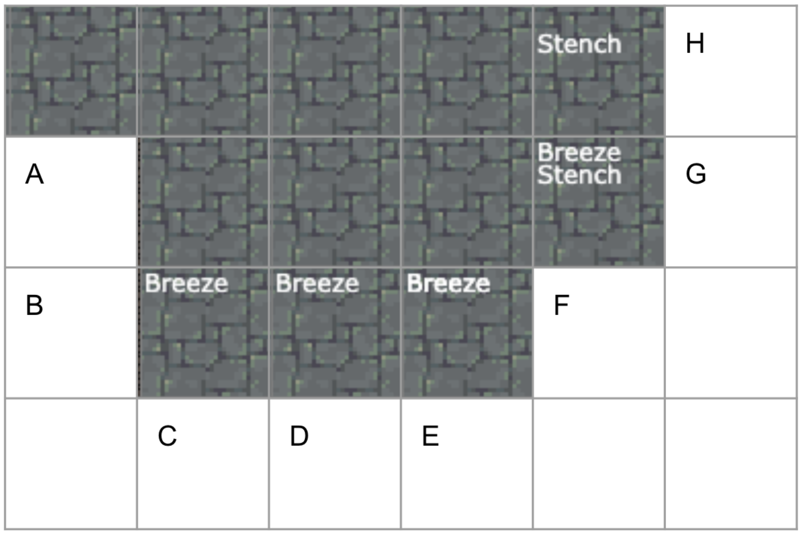


Figure 1: Safe, not safe, or unsure?

For the squares A-H, mark each of them with ‘+’ if the square is definitely safe, ‘–’ if it is definitely not safe, and ‘?’ otherwise.

(b) Consider the PL symbols $S_A, S_B \ldots S_H$ that represent whether or not each square A-H is safe or not safe. We also have symbols W_i and P_i for whether square i has a wumpus or a pit, respectively.

Let our Knowledge Base (KB) be a propositional logic sentence in CNF which contains all of the logic for the rules of Wumpus World as well as the percepts for the squares already visited (i.e., info about stenches, breezes, or not in the 12 dark squares in the figure). Assume the KB also has logic related to the safety of a square, i.e.,$\neg W_i \wedge \neg P_i \iff S_i$.

You are given access to a black box SAT solver function `PL_SATISFIES(pl_sentence}` which returns a PL model satisfying the given sentence *if* it is satisfiable and `None` otherwise.

Write pseudocode using the SAT solver and given propositional logic symbols to determine is square i is:

(a) Definitely safe

(b) Definitely not safe

(c) Unsure

(c) 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 ← {[x, y] : ASK(KB, $OK^t_{x,y}$) = true}
  if ASK(KB, $Glitter^t$) = true then                                      A
    plan ← [Grab] + PLAN-ROUTE(current, {[1,1]}, safe) + [Climb]
  if plan is empty then                                                    B
    unvisited ← {[x, y] : ASK(KB, $L^{t'}_{x,y}$) = false for all $t' \leq t$}
    plan ← PLAN-ROUTE(current, unvisited ∩ safe, safe)
  if plan is empty and ASK(KB, $HaveArrow^t$) = true then                  C
    possible_wumpus ← {[x, y] : ASK(KB, ¬ $W_{x,y}$) = false}
    plan ← PLAN-SHOT(current, possible_wumpus, safe)
  if plan is empty then   // no choice but to take a risk                  D
    not_unsafe ← {[x, y] : ASK(KB, ¬ $OK^t_{x,y}$) = false}
    plan ← PLAN-ROUTE(current, unvisited ∩ not_unsafe, safe)
  if plan is empty then                                                    E
    plan ← PLAN-ROUTE(current, {[1, 1]}, safe) + [Climb]
  action ← POP(plan)
  TELL(KB, MAKE-ACTION-SENTENCE(action, t))
  t ← 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 ← ROUTE-PROBLEM(current, goals, allowed)
  return A*-GRAPH-SEARCH(problem)
```

Figure 2: Hybrid-Wumpus-Agent from AIMA 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.

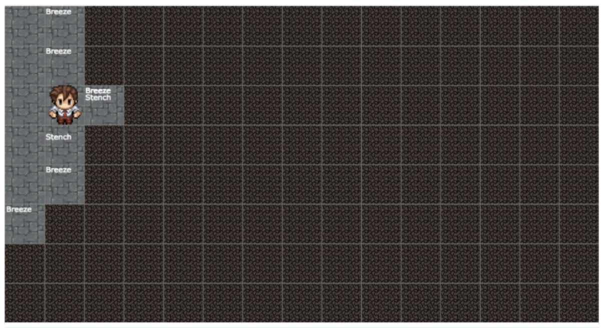
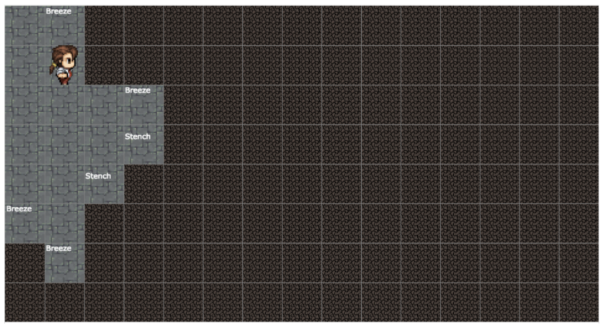
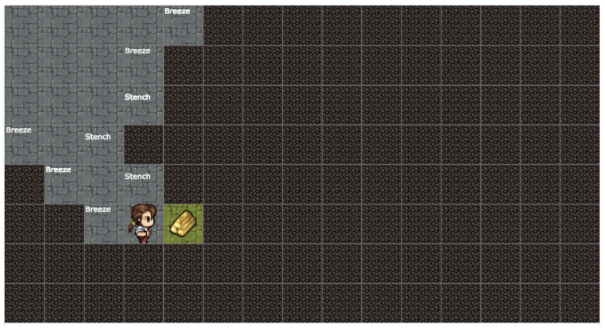
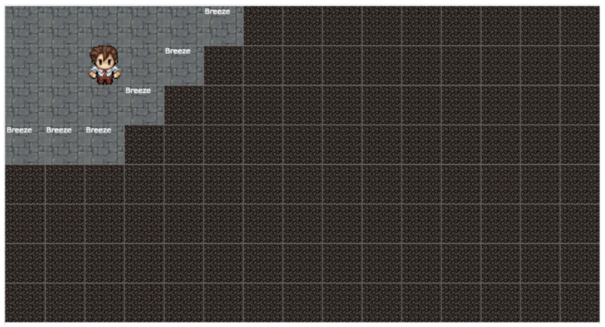
State	Code Chunk
	
	
	
	

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.

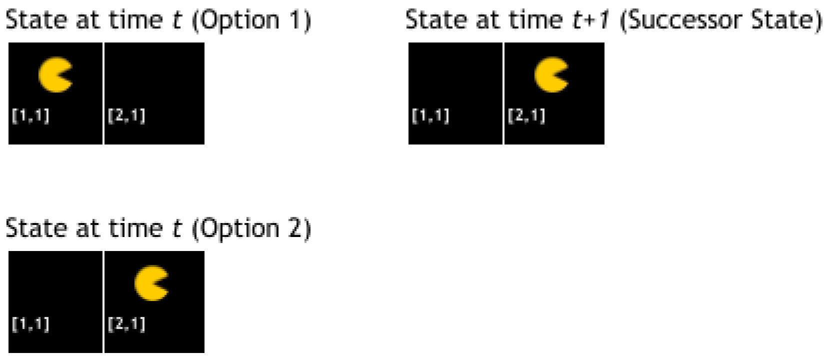

State at time *t* (Option 1)
[1,1]
[2,1]
State at time *t+1* (Successor State)
[1,1]
[2,1]
State at time *t* (Option 2)
[1,1]
[2,1]

Figure 3: Mini Pacman Grid

(c) Suppose that at time 0, Pacman is somewhere on a 5x5 grid ((1,1) at the bottom left, (5,5) at the top right) with only walls on the borders.

For each of the following, state whether the entailment relation is correct. Explain your reasoning.

(a) $Up^t \vee Right^t \models \neg Pacman^{t+1}_{(1,1)}$

(b) $\neg Pacman^{t+1}_{(1,1)} \models Up^t \vee Right^t$

(c) $Up^0 \wedge Up^2 \wedge Up^3 \models Pacman^4_{(x,y)} : x \in [1,5], y \in [4,5]$

(d) $Up^t \wedge Right^t \models \neg Pacman^{t+1}_{(5,5)}$

(e) $\neg Pacman_{(5,5)}^{t+1} \models Up^t \wedge Right^t$

(f) $Down^{t+1} \wedge Left^{t+1} \models Up^t \wedge Right^t$