1 Conceptual Review

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

- Symbols:

  Variables that can be T/F (capital letter)

- Operators:

  And ($\wedge$), Or ($\vee$), Implies ($\Rightarrow$), Equivalent ($\Longleftrightarrow$)

- Sentences:

  Symbols connected with operators, can be T/F

- Equivalence:

  True in all models that A and B imply each other (A equivalent to B)

- Literals:

  Atomic Sentence

- Knowledge Base:

  Sentences known to be true

- Entailment:

  A entails B iff for every model that satisfies A, B is also true

- Clauses (Definite vs. Horn Clauses):

  Clause: A conjunction of literals
  Definite Clause: Clause with exactly one positive literal
  Horn Clause: Clause with at most one positive literal

- Model Checking:

  Check if sentences are true in given model/check entailment

- Theorem Proving:

  Search for sequence of proof steps (e.g. Forward Chaining)

- Modus Ponens:

  From P and $(P \Rightarrow Q)$, infer Q

(b) Recall the definitions of satisfiability and entailment.

- **Satisfiability**:

  A sentence is *satisfied by* some model (an assignment of values to variables) m if m makes the sentence true.
  A sentence is satisfiable if there exists a model that satisfies it.

- **Entailment**:

Entailment: $a \models b$ ("a entails b" or "b follows from a") iff every model that satisfies a also satisfies b. In other words, the a-worlds (worlds where a is true) are a subset of the b-worlds [$models(a) \subseteq models(b)$].

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

Satisfiability holds if there exists a single model that in which the sentence is true. This is a property of a sentence (a sentence is satisfiable or it is not).

Entailment holds if all models which satisfy one sentence (query) also satisfy another sentence (knowledge base). It relates two sentences (sentence a entails b, or a does not entail b), and requires checking all models that satisfy a.

(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

(b) $A \Rightarrow B$
By definition of implication, in all models (i.e., truth assignments) where A is true, B is also true. Thus in all models, $A \Rightarrow B$ is satisfied.
(Thinking of it conversely, there would never be a model where B is false and A is true. By definition of implication, this means no model the rule $A \Rightarrow B$.)

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

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

  False (when A is true and B is False, $A \vee B$ is true but $A \Rightarrow B$ is false)

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

  True (the RHS is $B \Rightarrow A$, which is one of the conjuncts in the definition of $A \iff B$)

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

  True (the model has A and $\neg B$)

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

SAT can be modeled as a CSP in which the variables are literals with domain $\{\top, \bot\}$, and the constraints are the clauses themselves.

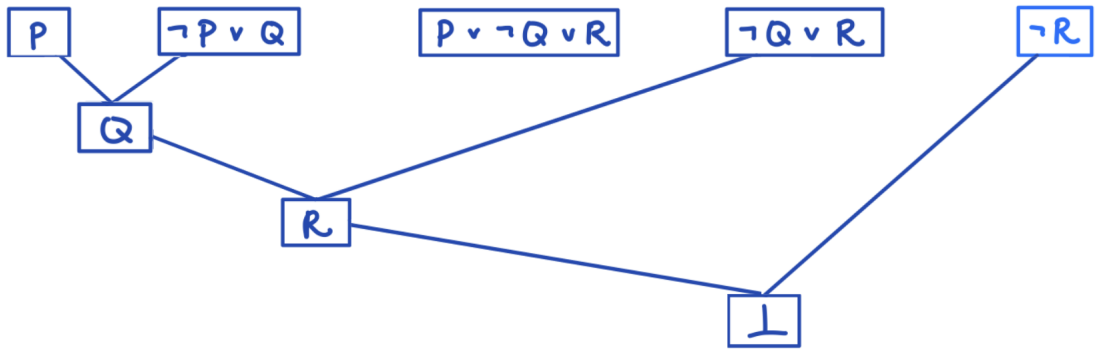
(g) Suppose we have an algorithm which determines whether a sentence is satisfiable or not. Given two sentences A and B, how could we determine whether $A \models B$?

If $A \models B$, then $A \wedge \neg B$ should be unsatisfiable (this is proof via reductio ad absurdum - reduction to an absurd thing).

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:

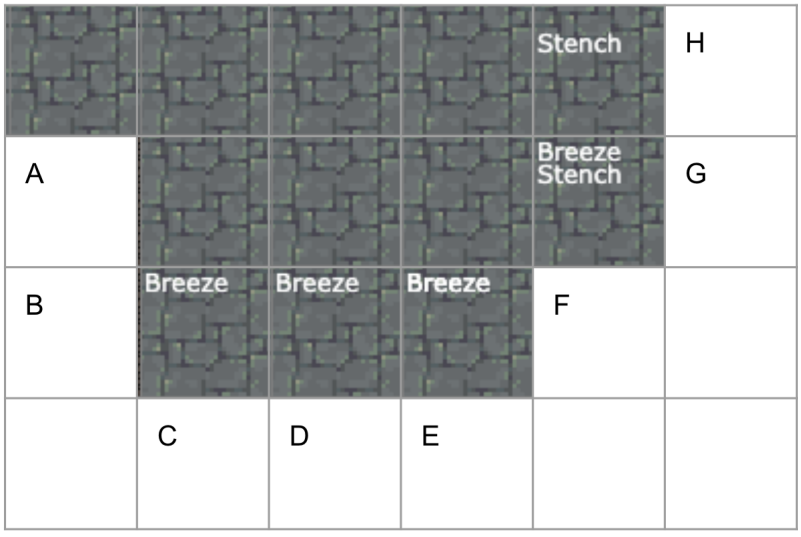
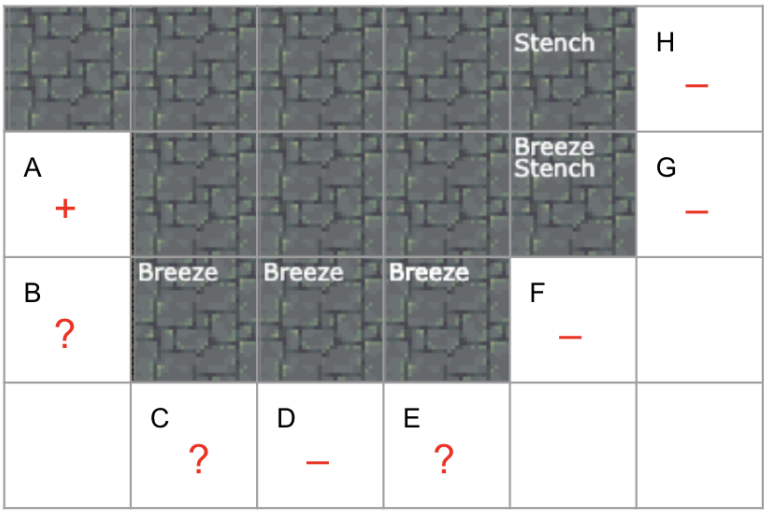


Figure 1: Safe, not safe, or unsure?

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



(a) Safe, since there is no adjacent breeze or stench.

(b) Unsure, since there may be a pit.

(c) Unsure, since there may be a pit.

(d) Not safe - it's the only square adjacent to the breeze, so it must be a pit.

(e) Unsure, since there may be a pit.

(f) Unsafe, there are two squares adjacent to the breeze/stench square, so both must be hazards.

(g) Unsafe - same reasoning as F.

(h) Unsafe. Since there is only one square adjacent to the stench, this square is a wumpus.

(b) Consider the PL symbols $S_A, S_B \dots S_H$ 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

```
if PL_SATISFIES(KB and not S_i) == None:
    return SAFE
```

(b) Definitely not safe

```
if PL_SATISFIES(KB and S_i) == None:
    return SAFE
```

(c) Unsure

```
if PL_SATISFIES(KB and S_i)  != None
&& PL_SATISFIES(KB and !S_i) != None:
    return 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
$safe \leftarrow \{[x, y] : \text{ASK}(KB, OK^t_{x,y}) = true\}$
if ASK(KB, $Glitter^t$) = *true* **then** A
  $plan \leftarrow [Grab] + \text{PLAN-ROUTE}(current, \{[1,1]\}, safe) + [Climb]$
if *plan* is empty **then**
  $unvisited \leftarrow \{[x, y] : \text{ASK}(KB, L^{t'}_{x,y}) = false \text{ for all } t' \leq t\}$ B
  $plan \leftarrow \text{PLAN-ROUTE}(current, unvisited \cap safe, safe)$
if *plan* is empty and ASK(KB, $HaveArrow^t$) = *true* **then**
  $possible\_wumpus \leftarrow \{[x, y] : \text{ASK}(KB, \neg\, W_{x,y}) = false\}$ C
  $plan \leftarrow \text{PLAN-SHOT}(current, possible\_wumpus, safe)$
if *plan* is empty **then** // no choice but to take a risk
  $not\_unsafe \leftarrow \{[x, y] : \text{ASK}(KB, \neg\, OK^t_{x,y}) = false\}$ D
  $plan \leftarrow \text{PLAN-ROUTE}(current, unvisited \cap not\_unsafe, safe)$
if *plan* is empty **then**
  $plan \leftarrow \text{PLAN-ROUTE}(current, \{[1, 1]\}, safe) + [Climb]$ E
$action \leftarrow \text{POP}(plan)$
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 ← 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.

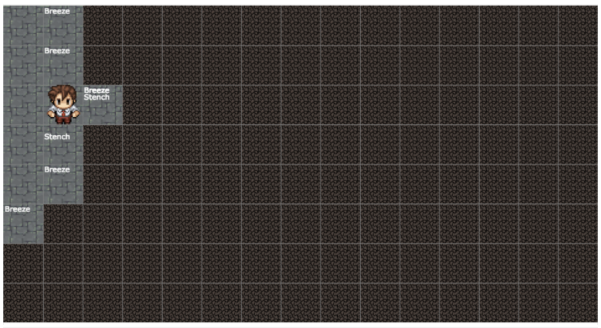
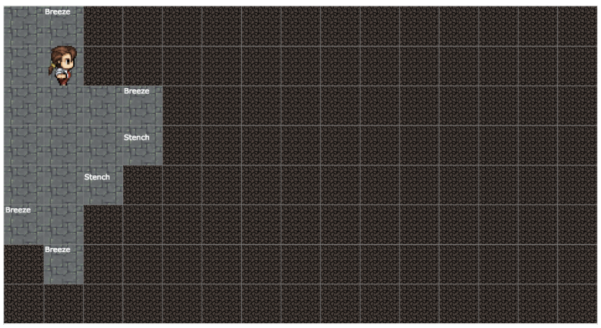
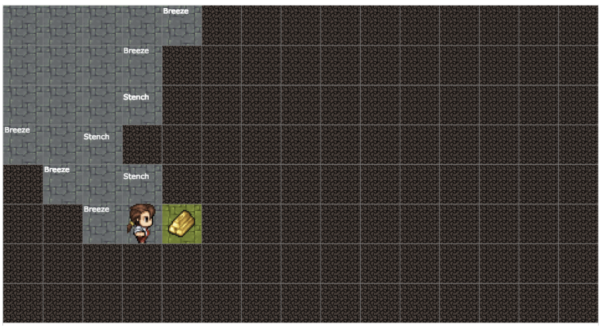
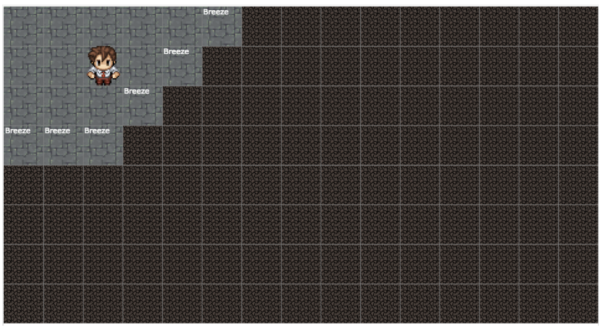
State	Code Chunk
	C: There are two stenches within reasonable distance; therefore, based on satisfiability, it is possible for a Wumpus to exist in the unvisited square diagonal from our explorer. There are also no guaranteed safe spaces. Since we have an arrow that we can use in case of Wumpus, we plan to shoot our arrow.
	B: We have found an unvisited square free from breeze or stench. We know, based on our knowledge base, that this unvisited square must therefore also be safe and we can visit it.
	A: We have found gold, and we can grab it, and then plan the shortest, safest route out.
	D: There is no guaranteed safe square; therefore, we must take a risk.

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?

Successor-state axioms are axioms outlining what preconditions must be true in order to ensure that the state at the next time step will be specified. By definition, it is an axiom that sets the truth value of F^{t+1} (where F is some fluent, or changeable variable in an environment) in one of two ways:

- The action at time t causes F to be true at $t+1$ (which refers to $ActionCausesF^t$)
- F was already true at time t and the action at time t does not cause it to be false.

It has the following schema: $F^{t+1} \iff ActionCausesF^t \vee (F^t \wedge \neg ActionCausesNotF^t)$.
We use successor-state axioms to ensure that each state we compute is the result of legal action.

(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.

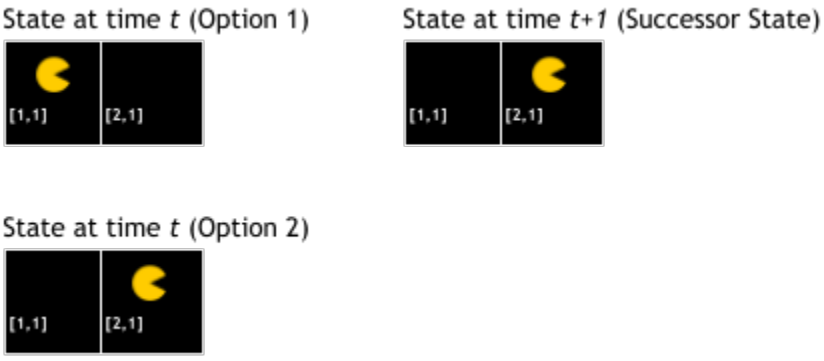


Figure 3: Mini Pacman Grid

Successor-state axiom: $Pacman^{t+1}_{(2,1)} \iff Right^t \vee (Pacman^t_{(2,1)} \wedge \neg Left^t)$

F^{t+1} is $Pacman^{t+1}_{(2,1)}$
$ActionCausesF^t$ is $Right^t$
$(F^t \wedge \neg ActionCausesNotF^t)$ is $(Pacman^t_{(2,1)} \wedge \neg Left^t)$

Think about how you could prevent Pacman from being in multiple states or taking multiple actions at the same time. You will get to explore this in P3!

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

True, there is no square that would lead to square (1,1) after moving up or right

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

False, a counterexample would be starting at square (3,2), and an action left leading to square (2,2)

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

False, this is almost true, however, if Pacman starts at row 1 and the action at step 1 was down, Pacman would end at row 3

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

True

There is no model that fits the action at a time step being both Up and Right. Therefore, for every model that fits this, the right side must also be true

(similar to being vacuously true for implications)

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

False, since there is no model such that the right side is true, and there is at least one model such that the left side is true

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

True, since there are no valid models in the left or the right

(similar to False $\implies$ False)