Recitation 4 September 20

1 Discussion-Based Warm Ups

(a) Determine which of the following are correct, and explain your reasoning:

(i) $(A \vee B) \models (A \implies B)$

(ii) $A \iff B \models A \vee \neg B$

(iii) $(A \vee B) \wedge \neg(A \implies B)$ is satisfiable.

(b) What is the difference between satisfiability and entailment (think about the purpose and requirements of each)?

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 ← [*Grab*] + PLAN-ROUTE(*current*, {[1,1]}, *safe*) + [*Climb*]
  if *plan* is empty **then** B
    $unvisited \leftarrow \{[x, y] : \text{ASK}(KB, L^{t'}_{x,y}) = false \text{ for all } t' \leq t\}$
    plan ← PLAN-ROUTE(*current*, *unvisited* ∩ *safe*, *safe*)
  if *plan* is empty and ASK(KB, $HaveArrow^t$) = *true* **then** C
    $possible\_wumpus \leftarrow \{[x, y] : \text{ASK}(KB, \neg\, W_{x,y}) = false\}$
    plan ← PLAN-SHOT(*current*, *possible_wumpus*, *safe*)
  if *plan* is empty **then** // no choice but to take a risk D
    $not\_unsafe \leftarrow \{[x, y] : \text{ASK}(KB, \neg\, OK^t_{x,y}) = false\}$
    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))
  $t \leftarrow t + 1$
  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 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.

2 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 back 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!

(a) Consider the following Wumpus World state:

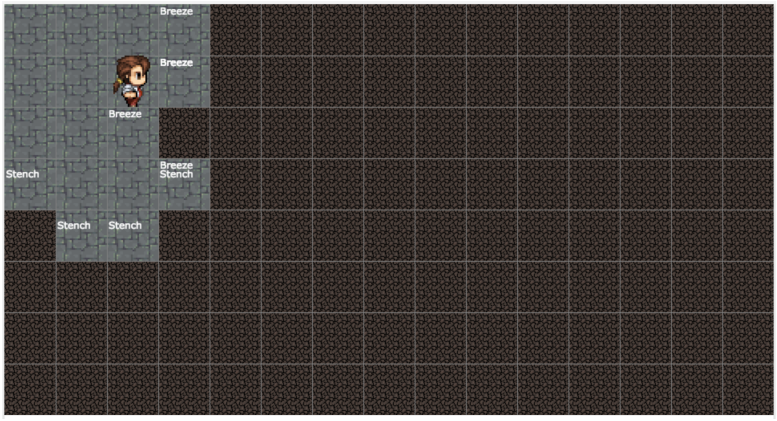

Figure 2: Entailment versus Satisfiability?

Based on our previous discussion around entailment and satisfiablity, 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) Now, refer to Figure 2 from Page 2, and take a moment to familiarize yourself with the pseudocode 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.

Match each of the following states to one of the labeled code chunks in the pseudocode, and explain your reasoning.

Figure 3: Which code chunk is applicable for each of these states?

3 Axioms & Arrows

Up until now we have assumed that the plans we create always make sure that an actions preconditions are satisfied. Let us now investigate what propositional successor-state axioms such as $HaveArrow^{t+1} \iff (HaveArrow^t \wedge \neg Shoot^t)$ have to say about actions whose preconditions are not satisfied.

(a) First, let us consider what successor-state axioms are. How do they differ from action axioms, and why might we choose to use them?

(b) Show that the axioms predict that nothing will happen when an action is executed in a state where its preconditions are not satisfied.

(c) Consider a plan p that contains the actions required to achieve a goal but also includes illegal actions. Is it the case that successor-state axiom will allow the actions?

We recommend that you write a truth table and ask yourself the following question when looking at the truth table:

- Can I shoot if I don't have an arrow?

4 SAT and DPLL

Recall the Davis-Putnam-Logemann-Loveland (DPLL) algorithm from lecture. DPLL conducts backtracking search over possible models (i.e., assignments to all variables) with these additional checks:

- Return true if all clauses are satisfied (we have found the answer)
- Return false if any clause is falsified (need to backtrack).
- If all occurences of a symbol in as-yet-unsatisfied clauses have the same sign, then give the symbol that value; e.g., in $(\neg A \vee B) \wedge (\neg A \vee \neg C)$, set A to false, which satisfies all the remaining clauses.
- If a clause is left with a single literal, set the literal to satisfy the clause; e.g., in $(B) \wedge (\neg B \vee \neg C)$, set B to true to satisfy the left clause. This often leads to new unit clauses, as in this example where $\neg C$ will be in a clause by itself.

(a) What is the difference between truth table entailment and solving SAT with DPLL?

(b) Find if the following is satisfiable. Other than the above rules, assign values to symbols alphabetically, and begin with symbol = true.

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

5 Resolution

Resolution

Algorithm Overview

function PL-RESOLUTION?(KB, α) returns true or false

We want to prove that KB entails α

In other words, we want to prove that we cannot satisfy (KB and **not** α)

1. Start with a set of CNF clauses, including the KB as well as $\neg\alpha$
2. Keep resolving pairs of clauses until
   - A. You resolve the empty clause
     - Contradiction found!
     - KB $\wedge \neg\alpha$ cannot be satisfied
     - Return true, KB entails α
   - B. No new clauses added
     - Return false, KB does not entail α

From the knowledge base below, prove $\neg P_{1,2}$. Note that this is the same example as in Lecture 7. We did not have time to step through the algorithm in lecture and we'll do that now!