Logical Agents

CS156

Chris Pollett

Oct 6, 2014

Outline

- Knowledge-Based Agents
- Quiz
- Wumpus World Example
- Logic

Introduction

- People know stuff; knowing stuff help you do things.
- We next study the process of **reasoning** that operates on some internal **representation of knowledge**.
- Agents that do this kind of reasoning are called **knowledge-based agents**.
- Before we begin, let's briefly consider why problem-solving agents can't do this very well.
- Consider the 8-puzzle: Knowledge of what each action does is very limited, its score tells us something limited about how well an action might lead to a solution of the puzzle.
- But we cannot deduce from this limited knowledge things like two squares can't occupy the same space, or that states with odd parity can't be reached from states with even parity.
- CSPs allow us to represent states in more complex, non-atomic ways, but are limited to just the relations of the problem.
- To increase our reasoning power, we use various kind of **logics** which extend simple relations.
- To start let's formulate a broad design for a knowledge-based agent.

Knowledge-based agents

- The central component of a knowledge-based agents is a knowledge-base (KB).
- We should consider this as a collection of **sentences** -- we will formally define sentence in a moment -- which are true about the world.
- Sentences will be expressed in a **knowledge representation language**.
- Sometimes we will call a sentence an **axiom**, when the sentence is taken as given without being derived from other sentences.
- Often knowledge bases are very similar to databases.
- Two things we want to be able to do with a knowledge base are:
  1. ***Tell*** *it new facts (update/tell process)*
  2. ***Ask*** *questions about the world (query)*
- Both of these operations may involve **inference** -- that is, deriving new sentences from old. Inference should obey the property that when we Ask a question, the inferred answer should be based on the Tell operations performed on the KB system.

KB-Agent Pseudo-code

Below is pseudo-code for a knowledge-based agent. As with all of our agents, it receives percepts about its environment, and returns actions. It maintains a KB, which initially may contain some **background knowledge**.

```
function KB-Agent(percept) returns an action
    persistent KB, a knowledge base
    t, a counter, initially 0, indicating time

    // store the percept into the KB
    Tell(KB, Make-Percept-Sentence(percept, t))

    // choose an action
    action := Ask(KB, Make-Action-Query(t))

    // tell the KB, we did that action at time t
    Tell(KB, Make-Action-Sentence(action, t))

    t++
    return action
```

Remarks on KB Agent

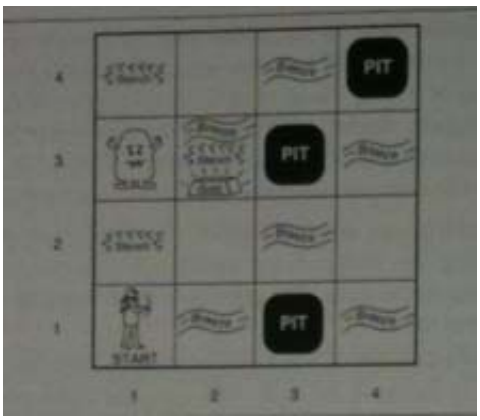
- Because of Tell and Ask, the KB-agent is not an arbitrary program for calculating actions.
- We can predict the behavior of a KB-agent at the **knowledge level**, that is, we know what it will do based on its goals and what it knows about the world.
- For example, an automated taxi program might have the goal of taking a passenger from SF to Marin County, and it might know that the Golden Gate Bridge is the only link between these two locations.
- We can predict that the automated taxi will cross the Golden Gate Bridge, because it knows that that will achieve its goal.
- We know this independent of how the taxi works at the implementation level.
- KB agents can be built in two ways: One can start from an empty KB and simply tell the agent all the things it needs to know. This is called the **declarative approach**. Or one can try to encode the desired behaviors directly in the program code. This is called the **procedural approach**.

Quiz

Which of the following is true?

1. Node consistency of a network of constraints ensures that constraints in a CSP involving single variables are all satisfied
2. Path consistency of a CSP is satisfied if a CSP is strongly 1-consistent.
3. The MRV heuristic for CSP solving says pick the value that rules out the fewest choices for the neighboring variables in the constraint graph.

Wumpus World



- We next describe an environment in which a KB-agent might be useful.
- The **wumpus world** is a cave consisting of rooms connected by passageways.
- In this cave lives a wumpus which eats anyone that enters the room that it is in. The wumpus doesn't move between rooms.
- The wumpus can be shot by the agent, but the agent only has one arrow.
- Some rooms in this world contain bottomless pits that trap anyone who enters these rooms, except a wumpus.
- Some rooms might have a heap of gold.

PEAS Description of Wumpus World

- **Performance measure**: +1000 for climbing out of the cave with gold. - 1000 for falling into a pit or being eaten by a wumpus. -1 for each action taken and -10 for using the arrow. The game ends when the agent dies or when the agent climbs out of the cave.
- **Environment**: A 4x4 grid of rooms. The agent always starts in the square labeled [1,1], facing to the right. The location of the gold and the wumpus are chosen randomly, with a uniform distribution, from squares other than the start square. In addition, each of these other squares has a pit with a 0.2 probability.
- **Actuators**: The agent can move forward, turn left 90 degrees, turn right 90 degrees, grab gold (if gold in the same square), shoot (only once, arrow goes in direction facing), or climb (if in exit square [1,1]). The agent dies if it enters a room with a pit or a non-dead wumpus. If an agent moves forward into a wall it does not move.
- **Sensors**: The agent has five sensors: Stench (is on if Wumpus within one square of player), Breeze (is on if player within one square of pit), glitter (if in a

room with gold), Bump (if walks into wall), Scream (if wumpus is killed). Percepts will be given as a list. For example, [Stench, Breeze, None, None, None].

Notes about Wumpus World

- The wumpus world is discrete, static, and single-agent.
- It is sequential, because rewards may come only after many actions are taken.
- It is partially observable.
- The main challenge for an agent is overcoming its initial ignorance of the configuration of the environment.
- In most possible wumpus worlds, it is possible for the agent to successfully retrieve the gold. Only about 21% are unfair in that the gold is surrounded by pits or the like.
- Let's look at how a KB-agent might reason about the wumpus world

Wumpus World at Stages of Reasoning

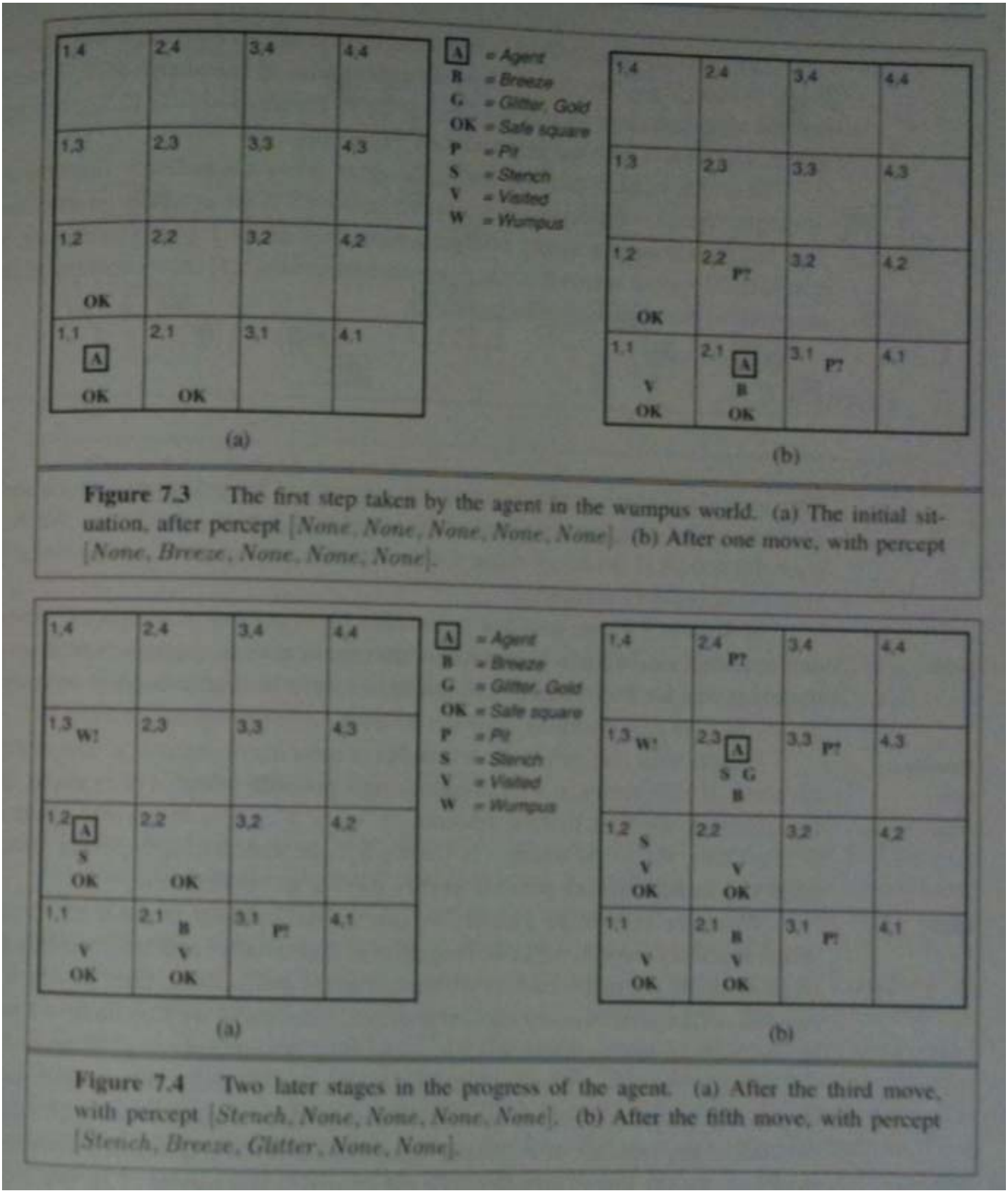


Figure 7.3 The first step taken by the agent in the wumpus world. (a) The initial situation, after percept [*None*, *None*, *None*, *None*, *None*]. (b) After one move, with percept [*None*, *Breeze*, *None*, *None*, *None*].

Figure 7.4 Two later stages in the progress of the agent. (a) After the third move, with percept [*Stench*, *None*, *None*, *None*, *None*]. (b) After the fifth move, with percept [*Stench*, *Breeze*, *Glitter*, *None*, *None*].

- Figure 7.3(a) illustrates the initial knowledge our KB-agent might have about the wumpus world.

- We denote that the player is in the square by writing an A; we denote that the agent knows the square is safe, by putting an OK in it.
- The first percept is [None, None, None, None, None], from which the agent can conclude thats its neighboring squares [1,2] and [2, 1] are OK.
- The agent moves to squares which it knows will be okay, if possible.
- Suppose it moves to [2,1]. The agent perceives a breeze so it tells its representation that [2,1] has a breeze. This means that it marks [2,2] and [3,1] with "P?" to indicate that these squares might have a pit.
- Our agent now takes to go one to the squares in its KB which is marked okay and is nearest. In this case, [1,2] is the only okay square left. So it moves [1,1], then, [1,2] to get there.
- Here it smells a stench, but doesn't detect a breeze. As it didn't detect a stench in [2,1], it knows that [2,2] doesn't have a wumpus. Therefore, the Wumpus must be in [1,3]. As it didn't detect a breeze in [1,2] it knows [2,2] does not have a pit, so is okay. This also, means that [3,1] must have had the pit.
- This show that logical reasoning can be very powerful.

Logic

- We said our knowledge base consist of sentences.
- These sentences are expressed using according to a **syntax**, which specifies all the sentences that are well-formed.
- For example, something like $x + y = 4$ is a well-formed formula, but $x4y+=$ is not.
- A logic must also define the **semantics** or meaning of sentences.
- The semantics defines the **truth** of each formula with respect to each **possible world**.
- For example, $x + y = 4$ is true in a world where x is 2 and y is 2, but false in a world where x is 1 and y is 1
- We often use the word **model** in place of "possible world".
- We say things like m **satisfies** a to mean the formula a is true in model m. We also say m is a model of a. We write $M(a)$ for all models of a.

Entailment

- We have already said semantics tells us about the truth of a syntactically correct statement in a possible world/model
- We now want to consider logical reasoning. To do this we begin by looking at **entailment** between sentences -- the idea that one sentence follows from another sentence/set of sentences.
- We say $\alpha \models \beta$ iff $M(\alpha) \subseteq M(\beta)$. We read $\models$ as entails
- For example, if we considers worlds satisfying the Wumpus World rules, the fact that we didn't feel a breeze at $(1,1)$ entails that there is no pit in $(1,2)$. On the other hand, when we felt a breeze in $(1,2)$ it didn't entail there was a pit in $(1,1)$.
- The process of using entailment to derive conclusions is called **logical inference**.
- One way to infer $KB \models \alpha$ is to check every single possible model of say $M(KB)$ and verify it is a model of α. This is called **model checking**
- General we would like to come with faster ways to determine entailment.
- We write $KB \vdash_i \alpha$ when we want to express that it was our inference algorithm i that derived α as following from KB.

- An inference algorithm that derives only entailed sentences is called **sound** or **truth-preserving**.
- An inference algorithm is **complete** if it can derive any sentence that is entailed.