5. Logical agents

By: Alex S.

Feb, 2024

1. Knowledge-based agents

This type of agents has a representation of knowledge. It is mostly defined as a knowledge base(KB). KB is described with the set of **sentences** or **axioms**.

There must be a way to add new sentences to the KB and a way to query what is known. The standard names for these operations are **TELL** and **ASK**, respectively. Both operations may involve **inference** - that is, deriving new sentences from old.

Each time the agent program is called, it does three things:

1. It **TELLs** the KB what it perceives.
2. It **ASKs** the KB what action it should perform.
3. The agent program **TELLs** the knowledge base which action was chosen, and the agent executes the action.

The procedure is shown in the diagram(Figure 1):

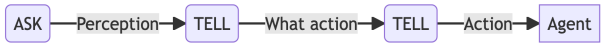

Figure 1: Inference mechanism

A generic knowledge-based agent can be described with the following pseudocode:

function KB-agent : $P \rightarrow a$ `is`
  let `KB be persistent`
  let $t \leftarrow 0$
  tell : $(\text{KB}, \text{do-percept-sentence} : (P, t))$
  ask : $(\text{KB}, \text{do-action-query} : t \rightarrow a) \rightarrow a$
  tell : $(\text{KB}, \text{do-action-sentence} : (a, t)) \rightarrow a$
  $t \leftarrow t + 1$
return a

where P - perception set; t - counter, time; a - action; KB - knowledge base

Agent's engineering can happen on different levels:

1. **Knowledge level** - where agent is being described with natural language - what it knows and what its goals are.
2. **Knowledge representation level** - How we represent the knowledge, e.g. semantic network, conceptual maps, object-oriented frames, logic language
3. **Implementation level** - how the agent is implemented and what tools are used for this goal.

2. The Wumpus World (Practical example)

Here we define the game where agent should achieve its given goal.

For the game we define the **PAGE** properties - *perceptions*, *actions*, *goals* and *environment*.

- **Goal** - to reach the gold ingot, take it and then come back to the starting position $(1, 1)$.
- **Performance measure** - is measured as a sum of points.
  - 100 points if the agent reaches the goal.
  - -100 points if the agent dies.
  - -1 points for every taken action.
  - -10 points for shooting an arrow into Wumpus.
- **Environment** - a 4x4 grid of rooms. The agent always starts at the square $(1, 1)$ facing to the right.

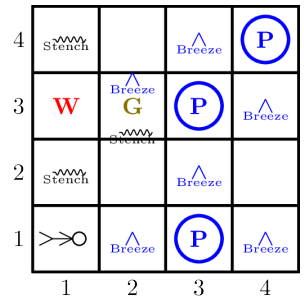

Figure 2 represents an example environment of generated randomly Wumpus World. Each initial state has *1 gold*, *1 Wumpus* and *3 pits*.

Figure 2: Wumpus World example

The initial state of the board is generated in a random manner, i.e. Wumpus and gold are chosen randomly with uniform distribution, excluding the starting square. Each other square can be a pit with the probability of 0.2. Probability number can be whatever, it can be a throw of a dice with the probability $\frac{1}{6}$. The gold can be in the same spot where *Wumpus* is, but cannot be in the same place where the pit is.

- **Sensors** - the agent has 5 sensors, each of which gives a single bit of information. The agent cannot percept diagonally.
  - The squares adjacent to the square that contains *Wumpus* has *Stench*. In Figure 2 square $(1, 3)$ emits stench to squares $(1, 2), (1, 4), (2, 3)$

- In the squares next to *pit*, the agent percepts *breeze*. In Figure 2, pit $(3, 1)$ emits breeze in $(2, 1), (3, 2), (4, 1)$
- In the square with the gold, it will perceive *Glitter*
- When the agent walks into a wall, it will perceive a *Bump*
- When the *Wumpus* is killed, it emits *Scream* that can be perceived anywhere in the cave.

- **Actuators** - The agent can:
  - Move *forward* where he looks
  - *Turn left* 90°
  - *Turn right* 90°
  - *Grab* the gold in the same square where the agent is
  - *Shoot* the arrow that is fired in a straight line, in the direction the agent is facing.

Let's explore the world.

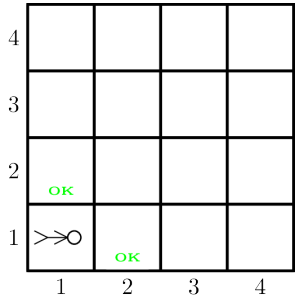

Figure 3: First move in the Wumpus World

The initial knowledge says that square $(1, 1)$ is safe, so it is marked as *OK*. The first percept is *[None, None, None, None, None]*(i.e. *[Stench, Breeze, Glitter, Bump, Scream]*). Therefore squares $(1, 2)$ and $(2, 1)$ are safe as shown in Figure 3.

Since both $(1, 2)$ and $(2, 1)$ are equally safe, it does not matter which to choose to go first. In the computer world, the first equal solution would be chosen. Let's imagine that the agent has chosen the square $(2, 1)$ and it moves forward$(100 - 1 = 99$ points, assuming it will bring the gold).

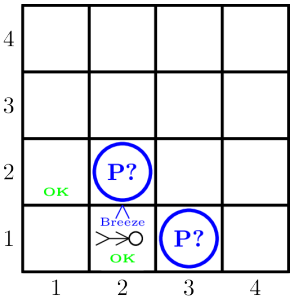

Figure 4: First move in the Wumpus World

The next percept is *[None, Breeze, None, None, None]*. The agent feels the breeze. The pit cannot be in square $(1, 1)$ by the rules. Therefore the pit or both can be on squares $(3, 1)$ or/and $(2, 2)$ as in Figure 4.

The notation **P?** indicates a possible pit in the square. The only safe square the agent knows is $(1, 1)$. So, it turns left 2 times, moves forward, turns right and moves forward to square $(1, 2)(99 - 5 = 94p)$.

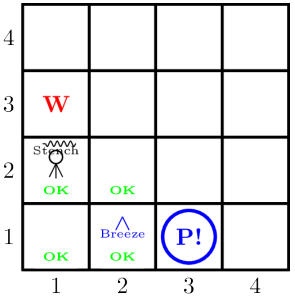

Figure 5: First move in the Wumpus World

In this state the agent percepts *[Stench, None, None, None]*. It means the *Wumpus* is nearby. But it cannot be in $(1, 1)$ square, and it cannot be in $(2, 2)$, otherwise it would have felt the stench in the square $(2, 1)$.

Therefore the agent can conclude that *Wumpus* is in $(1, 3)$. Moreover, the lack of breeze also indicates that there is no pit in $(2, 2)$ and it must be in $(3, 1)$ indicated by **P!** in Figure 5. The agent can infer that it is safe to move to the square $(2, 2)$, by rotating right and going forward$(94 - 2 = 92p)$. In $(2, 2)$ agent does not percept anything *[None, None, None, None, None]*. So, it is safe to move to $(2, 3)$ or $(3, 2)$. Let's consider the agent moves to $(2, 3)$ by rotating left and moving forwards$(92 - 2 = 90p)$.

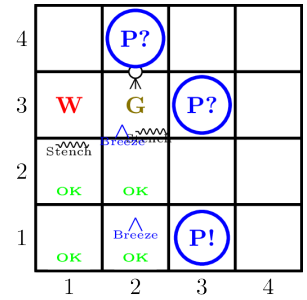

Figure 6: First move in the Wumpus World

In $(2, 3)$ the agent percepts *[Stench, Breeze, Glitter, None, None]*. So, it infers that possible pits are in $(3, 3)$ and $(2, 4)$, however, it grabs the gold as it feels the glitter$(90 - 1 = 89p)$ and starts creating a path that returns him back to $(1, 1)$.

For this purpose it needs to know the history of actions and use Dijkstra's algorithm or BFS for the shortest path. The actions to return home:

- Turn left 2 times$(89 - 2 = 87p)$
- Move to $(2, 2)(87 - 1 = 86p)$
- Turn right and move to $(1, 2)(86 - 2 = 84p)$
- Turn left and move to $(1, 1)(84 - 2 = 82p)$

In the end the agent returns back with the gold and 82 points.

3. Requirements for implementation of knowledge representation language

Languages can be split into 2 categories:

Artificially created languages

- E.g. programming, logic, mathematical languages

Natural languages

- E.g. languages we speak in
- Can be used for a communication between humans

- Can be used for a precise algorithm description,
- They are unambiguous, i.e. no hidden description
- They are not dependent on the context
- They are unclear and ambiguous
- They are dependent on the context

Characteristics of knowledge representation language:

- Expressive
- Concise and concentrated
- Unambiguous
- Context independent
- Effective, i.e. use inference mechanisms

Any language is defined by

1. **Syntax** - symbol configuration that is used to build sentences, e.g.
   - $x + y = 8$ is syntactically *correct*
   - $x8y =$ is syntactically *incorrect*
2. **Semantics** - defines the truth of each sentence with respect to each **possible world**. Semantics can be applied with the **interpretation** - that defines a relationship between sentences and facts in the real world. E.g. $x + y = 8$ can be true or false. The idea of this statement depends on the world state.

   If $x = 4$ and $y = 4$ the sentence is true.

   If $x = 7$ and $y = 4$ the sentence is false.

The amount of world states can be calculated with the formula:

$$\text{Number of worlds}(N) = 2^N$$

where N is the number of sentences.

4. Logical reasoning

Model is representation of a "possible world". Possible worlds might be thought as real environments that the agent might or might not be in.

Model is a mathematical abstraction, that simply fixes the truth or falsehood of every relevant sentence.

If sentence α is true in model m, we say that m **satisfies** α or m **is a model of** α.

Notation: $M(\alpha)$ - set of all models of α.

Logical entailment between sentences - the idea that a sentence *follows logically* from another sentence:

$$\alpha \vDash \beta$$

where α and β are sentences.

Entailment means that in every model in which α is true, β is also true.

$$\alpha \vDash \beta \text{ if and only if } M(\alpha) \subseteq M(\beta)$$

where $M(\alpha)$ is a subset of $M(\beta)$.

If $\alpha \vDash \beta$, then α is *stronger* assertion than β: it rules out more possible worlds. E.g. the sentence $x = 0$ entails the sentence $xy = 0$. In any model where x is zero, it is the case the xy is zero too.

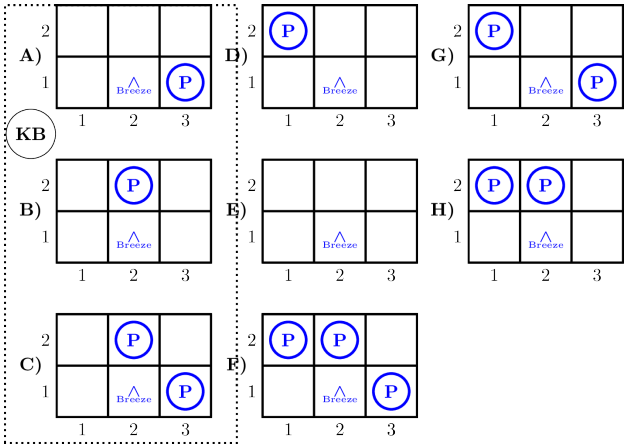

Figure 7: Possible models for the presence if pits in squares $(1, 2)$, $(2, 2)$ and $(3, 1)$. The KB corresponding to the observations of nothing in $(1, 1)$ and a breeze in $(2, 1)$ is shown by the dotted line.

In Figure 7 shown an example situation when the agent percepts a breeze in $(1, 2)$. The agent is interested in whether the adjacent squares $(1, 2)$, $(2, 2)$, and $(3, 1)$ contain pits. Each of the three squares might or might not contain a pit, so there are $2^3 = 8$ possible models.

The KB can be thought of as a set of sentences or as a single sentence that asserts all the individual sentences. The KB is false in models that contradict what the agent knows — for example, the KB is false in any model in which $(1, 2)$ contains a pit(D, F, G, H), because there is no breeze in $(1, 1)$. There are in fact just three models in which the KB is true, and these are shown in the dotted line in Figure 7.

Now let us consider 2 possible conclusions:

- α_1 = "There is no pit in $(1,2)$" which corresponds to states A, B, C and E in Figure 7
- α_2 = "There is no pit in $(2,2)$" which corresponds to states A, D, E and G in Figure 7

By inspection, we see the following: in every model in which *KB* is true, α_1 is also true. Hence, $\text{KB} \models \alpha_1$: there is not pit in $(1,2)$.

We can also see that in some models in which *KB* is true, α_2 is false. Hence, $\text{KB} \not\models \alpha_2$: the agent cannot conclude that there is no or there is a pit in $(2,2)$.

The example has shown that entailment can be applied to derive conclusions, i.e. carry out **logical inference**.

Model checking is the inference algorithm, shown in Figure 7 that enumerates all possible models to check that α is true in all models in which *KB* is true, so that $M(\text{KB}) \subseteq M(\alpha)$.

If an inference algorithm i can derive α from *KB*, we write:

$$\text{KB} \underset{i}{\vdash} \alpha$$

which is pronounced "α is derived from *KB* by i".

An inference algorithm that derives only entailed(true) sentences is called **sound** or **truth-preserving**, can also be called **deduction**.

The property of **completeness** is also desirable: an inference algorithm is complete if it can derive any sentence that is entailed.

Every **complete** inference can be **sound**, but not every **sound** inference can be **complete**.

The sentence is **valid** only if it is true in all possible interpretations and in all models, e.g.
- "Pit exists or pit doesn't exist".

The sentence is **executable**, if there is at least one interpretation in some model where the sentence is true, e.g.
- "Wumpus is in square $(1,4)$": might be true or false: $1 \vee 0 = 1$
- "Forward there is a wall and there is no wall": not **executable** as it is false in all models: $1 \wedge 0 = 0$

The correspondence between world and representation is shown in Figure 8.

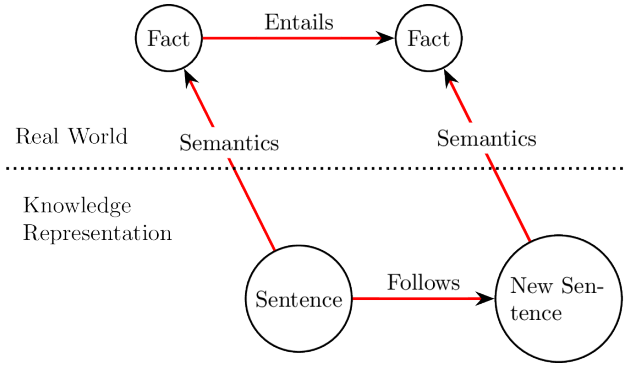

Figure 8: Sentences are physical configurations(aspects of the real world) of the agent, and reasoning is a process of constructing new physical configurations from old ones