Today's lecture:

- Knowledge based
- Propositional logic

Knowledge bases

- A set of sentences in a formal language than can TELL an agent what it needs to know.
- The agent can ASK its KB queries about what it should do

Architecture of a KB

- Inference engine: domain independent algorithms
- Knowledge base – domain specific content

An inference engine can be domain specific, for instance how to find shortest path between 2 cities in a KB that contains information about the map.

KB agent

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

  TELL(KB, MAKE-PERCEPT-SENTENCE(percept, t))
  action ← ASK(KB, MAKE-ACTION-QUERY(t))
  TELL(KB, MAKE-ACTION-SENTENCE(action, t))
  t ← t + 1
  return action
```

Given a percept – some perception about the world:

- It tells the KB about that thing it just perceived
- Then it asks the KB: given what I already knew and the new information (percept) – what action should I take?

Classic example: the Wumpus World

We have a grid, and in each cell a percept can be perceived:

- Percepts: breeze, glitter, smell
- Actions: move to different directions, grab, release, shoot

The goal is to obtaining a block of gold.

The gold creates glitter, the cells aside of pits are breezy, and there's a monster called the Wumpus that we move to its cell we die, but can fire an arrow on it to kill it. The cells around the Wumpus are "stinky".

Initially we don't know anything except what we percept about our current cell.

Exploring the Wumpus world is described in the slides.

- In the initial state: if we sense that our current cell is ok, we know up or left is ok – no pit/Wumpus.
- Along the way, the KB "remembers" what is the state in the cells we precieved/inferred about already, to query later.

Minesweeper is another example.

A simple KB for Wumpus World

Syntax: Boolean flags

- $P_{x,y}$ – true iff there's a pit in $[x, y]$
- $W_{x,y}$ – wumpus
- $B_{x,y}$ – breeze
- $S_{x,y}$ – smell

Semantics:

- No pit in [1,1]: $\neg P_{1,1}$
- $B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$
- Rules in the world: R_n, for rules specific to the world. For instance: $\neg B_{1,2}$

The goal is to provide new rules to the KB using inference.

Types of logic

- Propositional logic: based on facts, assigned true/false/unknown.
- First-order logic: facts, objects and relations, assigned true/false/unknown. Here we can define objects, like the wumpus, and say instead of the flag w_{23} something like $At(w, 2,3)$.
- Temporal logic: facts, objects, relations, times, assigned true/false/unknown. Every object has a time associated to it.
- Probability theory: facts, assigned degree of belief. Like: $w_{23} \sim 0.1$
- Fuzzy logic: degree of truth

Entailment

$KB \vDash \alpha$

KB entails a sentence α iff α is true in all worlds where the KB is strue.

Inference

$KB \vdash_i \alpha$

Sentence α can be inferred from KB by procedure i

Inference: enumeration method

To show that $KB \vDash \alpha$ we show that for all enumerations of the world (i.e. combinations of assignments to the participating variables), $KB = true$ infers $\alpha = true$, i.e.: $KB \Rightarrow \alpha$

This can be done using a truth table in small examples (like in the slides).

Simple inference in propositional logic

Simple **Modus Ponens**:

$$\frac{\alpha \Rightarrow \beta, \alpha}{\beta} \qquad \frac{\alpha \wedge \beta}{\alpha} \qquad \frac{\alpha \Leftrightarrow \beta}{\alpha \Rightarrow \beta \wedge (\beta \Rightarrow \alpha)}$$

Normal forms

- Conjunctive Normal Form – CNF: $(A \vee \neg B) \wedge (B \vee D)$
- Disjunctive Normal Form - DNF: $(A \wedge B) \vee (C \wedge A)$
- Horn Form: conjunction of Horn clauses, e.g. $(A \Rightarrow B) \wedge \big((C \wedge D) \Rightarrow A\big)$

Inference rules for propositional logic

Forward chaining:

The idea is to generate as many rules as possible using the given KB, and update it along the way, until the query is satisfied.

For instance, if we start with the KB:

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B

So we can start adding rules based on our current knowledge of the word, for instance (in order):

L

M

P

Q

And eventually we get to answering our query Q, which in this case is true.

Proof of completeness:

FC derives every atomic sentence that is entailed by KB:

1. FC reaches a fixed point where no new atomic sentences are derived
2. Let the final state be the model m, assigning true/false to symbols
3. Every clause in the original KB is true in m:

   $a_1 \wedge \ldots a_k \Rightarrow b$
4. Hence m is a model of KB

Backward chaining:

The idea is to work backwards from the query q. To prove q by BC:

- Check if q is known already. If not:
- Prove by BC all rules that imply q.

We need to avoid loops (check if a sub-goal is not already defined in the goal stack).

We also need to avoid repeating work (check every new sub-goal if it is already known)

Forward vs. backward chaining:

- FC - Forward: data-driven, automatic, unconscious processing. In most situations it would result with a lot of extra work irrelevant to the goal.
- BC - Backward: goal-driven, appropriate for problem solving.

BC complexity can be **much less than linear** in the size of KB.

Summary: look in the slides

Limits of propositional logic

We want to have a "breezy" literal (B_{xy}) for each rule that says that cells adjacent to pits are breezy, and that's inefficient.

First order logic

- Objects like in propositional logic, but also has:
- Relations: unary/n-ary, like $R_1(x), R_2(x,y,z)$ etc.
- Facts about some or all of the objects

The example from before would be represented by the fact: for all squares, if they are pits, their adjacent squares are breezy.

First order logic syntax

$\exists x \forall y Loves(x,y)$ – there is a person who loves everyone in the world

$\forall y \exists x Loves(x,y)$ – for everyone in the world, there exists someone who loves them

Syntax:

Atomic sentence:

- Predicate(term1, … term n)
- Term = function(term1,… ,term n) or a constant or a variable

Truth in FOL (semantics)

- Sentences are true w.r.t a **model** and an **interpretation**
- Model contains objects and relations among them
- Interpretation specifies referents for
    - Constants – objects
    - Predicate – relations
    - Functions – functional relations
- An atomic sentence $pred(term_1, \dots, term_2)$ is true iff the objects referred to by $term_1, \dots, term_n$ are in the relation referred to by the predicate.

  The relation is simply a set of n-vectors. So: $pred(x,y,z) = true \Leftrightarrow (x,y,z) \in R_{pred}$ where $R_{pred} = \{(x,y,z), \dots\}$