ARTIFICIAL INTELLIGENCE

LECTURE 6

Ph. D. Lect. Horia Popa Andreescu

2021-2022 3rd year, semester 5

- The slides for this lecture are based (partially) on chapter 7 of the Stuart Russel Lecture Notes [R, ch7], and on the same chapter from Russel & Norvig's book [RN, ch. 7]

KNOWLEDGE-BASED AGENT

- A Knowledge Base (KB) is, informally a set of sentences,
  - each sentence is expressed in a "knowledge representation language"
  - and it represents some assertion about the world.
- Features of a KB:
  - It is possible to add new sentences and to query what is known (the tasks are called TELL and ASK)
  - Deriving new sentences from old ones is called inference
  - The new sentences are based on the knowledge already known to the KB, which was TELLed to the KB previously

```
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(^))
   TELL(KB, MAKE-ACTION-SENTENCE(action, t))
   t ← t + 1
   return action
```

Figure 7.1 A generic knowledge-based agent.

The agent from Figure 7.1 [RN, 196] takes as input a knowledge (a percept) and returns an action.

The agent maintains a knowledge base (KB), initially composed of background knowledge.

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

1) It TELLS the knowledge base what it perceives.
2) It ASKS the knowledge base what action it should perform.
3) The agent records its choice with TELL and executes the action.

Declarative Vs. Procedural knowledge

- Designing the representation language to make it easy to express this knowledge in the form of sentences simplifies the construction problem enormously**.**
- This is called **the declarative approach** to system building.[RN 197]
- In contrast, the **procedural approach** encodes desired behaviors directly as program code; minimizing the role of explicit representation and reasoning can result in a much more efficient system. [RN 197]
- The agents can be provided with a mechanism of **learning** new facts by themselves.
- The new knowledge incorporated in the KB can be used for decision making, making the agent f**ully autonomous.**

THE WUMPUS WORLD EXAMPLE [RN, 197]

- The **wumpus world** is a cave consisting of rooms connected by passageways.
- Lurking somewhere in the cave is the wumpus, a beast that eats anyone who enters its room.
- The wumpus can be shot by an agent, but the agent has only one ar,row.
- Some rooms contain bottomless pits that will trap anyone who wanders into these rooins (except for the wumpus, which is too big to fall in).
- The only mitigating feature of living in this environment is the possibility of finding a heap of gold.
- This game is an excellent test bed for intelligent agents.

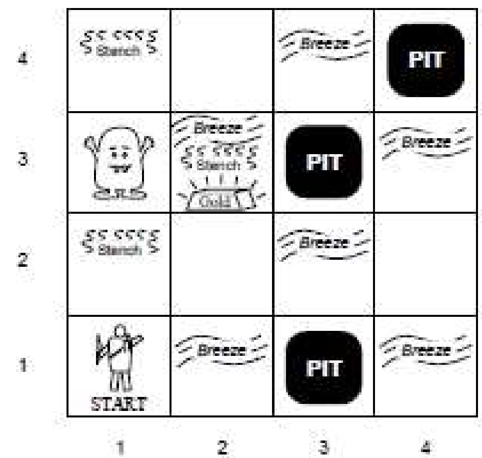
The wumpus world example (II) [R, 7/5]

Performance measure

- gold +1000, death -1000
- -1 per step, -10 for using the arrow

Environment

- Squares adjacent to wumpus are smelly
- Squares adjacent to pit are breezy
- Glitter iff gold is in the same square
- Shooting kills wumpus if you are facing it
- Shooting uses up the only arrow
- Grabbing picks up gold if in same square
- Releasing drops the gold in same square



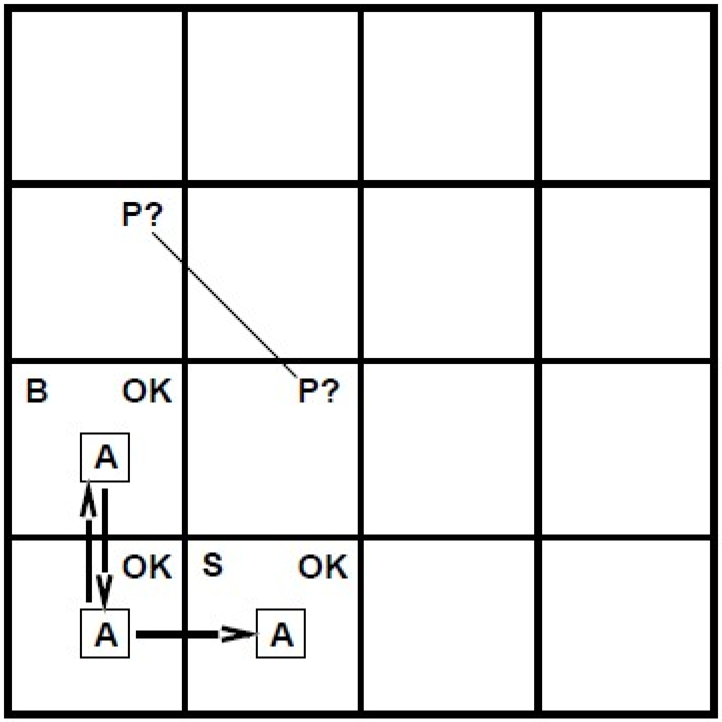
Actuators Left turn, Right turn,
Forward, Grab, Release, Shoot

Sensors Breeze, Glitter, Smell

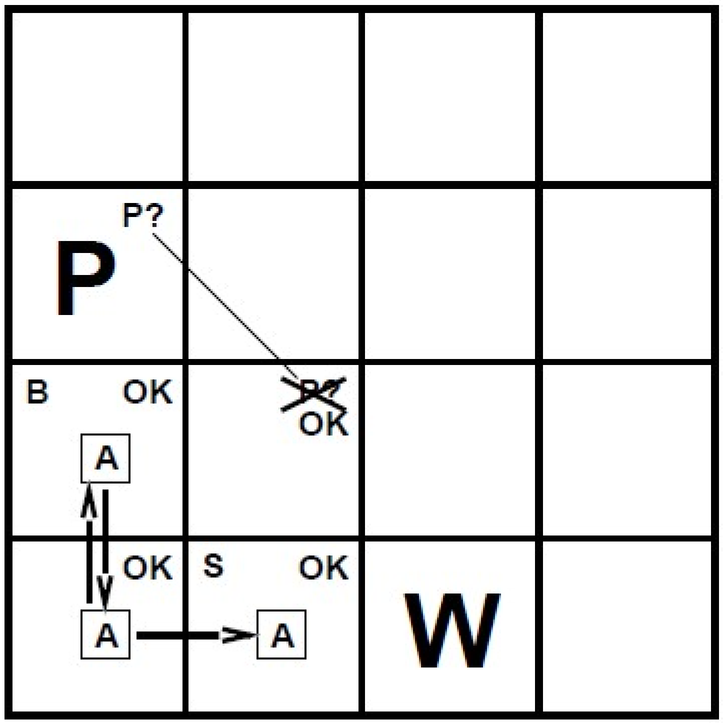
WUMPUS WORLD CHARACTERIZATION [R, 7/6]

- Observable?? No – only local perception
- Deterministic?? Yes – outcomes exactly speced
- Episodic?? No – sequential at the level of actions
- Static?? Yes – Wumpus and Pits do not move
- Discrete?? Yes
- Single-agent?? Yes – Wumpus is essentially a natural feature

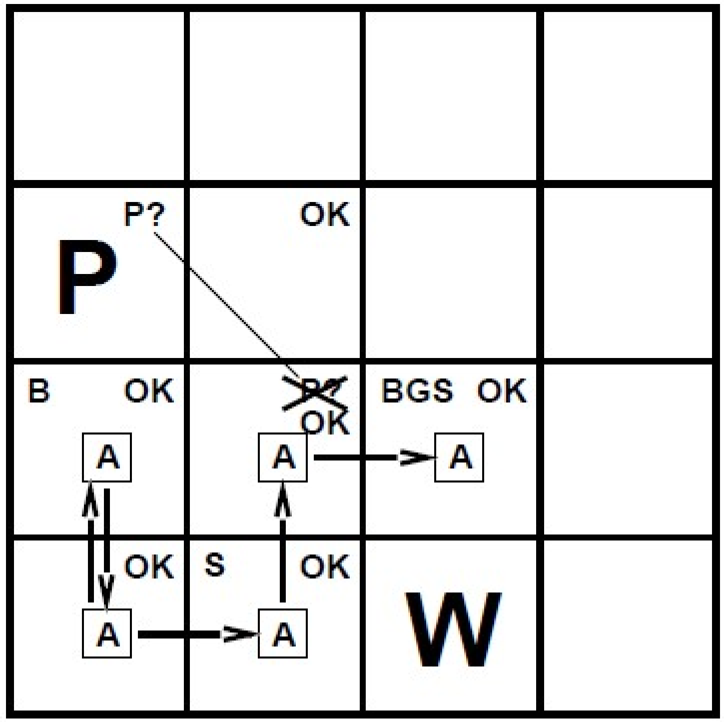
Exploring a wumpus world [R, 7/16]



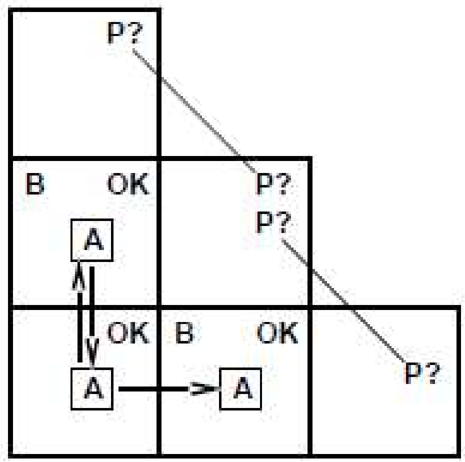
Exploring a wumpus world [R, 7/17]



EXPLORING A WUMPUS WORLD [R, 7/20]

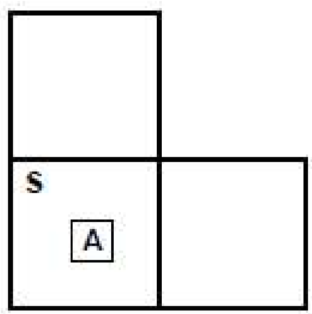


Reasoning in the Wumpus world [R, 7/21]



Breeze in (1,2) and (2,1)
$\Rightarrow$ no safe actions

Assuming pits uniformly distributed,
(2,2) has pit w/ prob 0.86, vs. 0.31



Smell in (1,1)
$\Rightarrow$ cannot move
Can use a strategy of coercion:
shoot straight ahead
wumpus was there $\Rightarrow$ dead $\Rightarrow$ safe
wumpus wasn't there $\Rightarrow$ safe

LOGIC [R, 7/22]

- Logics are formal languages for representing information such that conclusions can be drawn
- Syntax defines the sentences in the language
- Semantics define the “meaning" of sentences; i.e., define truth of a sentence in a world
- E.g., the language of arithmetic

  $x + 2 \geq y$ is a sentence; $x2 + y >$ is not a sentence

  $x + 2 \geq y$ is true i the number $x + 2$ is no less than the number y

  $x + 2 \geq y$ is true in a world where $x=7$; $y=1$

  $x + 2 \geq y$ is false in a world where $x=0$; $y=6$

ENTAILMENT [R, 7/23]

- Entailment means that one thing follows from another:
- KB ⊨ α
- Knowledge base KB entails sentence α

  if and only if

  α is true in all worlds where KB is true
- E.g., the KB containing \the Giants won" and \the Reds won"

  entails \Either the Giants won or the Reds won"
- E.g., x + y =4 entails 4=x + y
- Entailment is a relationship between sentences (i.e., syntax) that is based on semantics
- Note: brains process syntax (of some sort)

Models [R, 7/24]

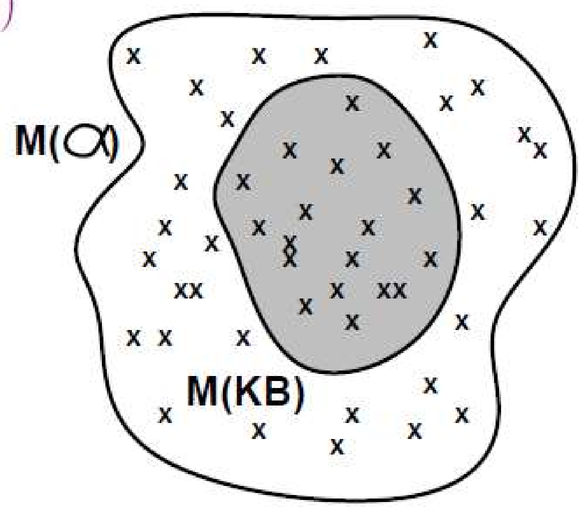
Logicians typically think in terms of models, which are formally structured worlds with respect to which truth can be evaluated

We say m is a model of a sentence α if α is true in m

$M(\alpha)$ is the set of all models of α

Then $KB \models \alpha$ if and only if $M(KB) \subseteq M(\alpha)$

E.g. KB = Giants won and Reds won
α = Giants won



INFERENCE [R, 7/31]

- KB $\vdash_i$ α = sentence α can be derived from KB by procedure I
- Consequences of KB are a haystack; α is a needle.
- Entailment = needle in haystack; inference = finding it
- Soundness: i is sound if

  whenever KB $\vdash_i$ α , it is also true that KB $\vDash$ α
- Completeness: i is complete if

  whenever KB $\vDash$ α , it is also true that KB $\vdash_i$ α
- Preview: we will dene a logic (rst-order logic) which is expressive enough to say almost anything of interest, and for which there exists a sound and complete inference procedure.
- That is, the procedure will answer any question whose answer follows from what is known by the KB.

Propositional Logic: Syntax [R, 7/32]

- Propositional logic is the simplest logic, it illustrates basic ideas
- The proposition symbols P1, P2 etc are sentences

If S is a sentence, ¬ S is a sentence (negation)

If S1 and S2 are sentences, S1 ^ S2 is a sentence (conjunction)

If S1 and S2 are sentences, S1 v S2 is a sentence (disjunction)

If S1 and S2 are sentences, S1 ⇒ S2 is a sentence (implication)

If S1 and S2 are sentences, S1 ⇔ S2 is a sentence (biconditional)

Propositional Logic: Semantics [R, 7/33]

Each model specifies true/false for each proposition symbol

E.g. $P_{1,2}$ $P_{2,2}$ $P_{3,1}$
$true$ $true$ $false$

(With these symbols, 8 possible models, can be enumerated automatically.)

Rules for evaluating truth with respect to a model m:

$\neg S$	is true iff	S	is false		
$S_1 \wedge S_2$	is true iff	S_1	is true **and**	S_2	is true
$S_1 \vee S_2$	is true iff	S_1	is true **or**	S_2	is true
$S_1 \Rightarrow S_2$	is true iff	S_1	is false **or**	S_2	is true
i.e.,	is false iff	S_1	is true **and**	S_2	is false
$S_1 \Leftrightarrow S_2$	is true iff	$S_1 \Rightarrow S_2$	is true **and**	$S_2 \Rightarrow S_1$	is true

Simple recursive process evaluates an arbitrary sentence, e.g.,
$\neg P_{1,2} \wedge (P_{2,2} \vee P_{3,1}) = true \wedge (false \vee true) = true \wedge true = true$

TRUTH TABLES [R, 7/37]

$B_{1,1}$	$B_{2,1}$	$P_{1,1}$	$P_{1,2}$	$P_{2,1}$	$P_{2,2}$	$P_{3,1}$	R_1	R_2	R_3	R_4	R_5	KB
false	*false*	*false*	*false*	*false*	*false*	*false*	*true*	*true*	*true*	*true*	*false*	*false*
false	*false*	*false*	*false*	*false*	*false*	*true*	*true*	*true*	*false*	*true*	*false*	*false*
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
false	*true*	*false*	*false*	*false*	*false*	*false*	*true*	*true*	*false*	*true*	*true*	*false*
false	*true*	*false*	*false*	*false*	*false*	*true*	*true*	*true*	*true*	*true*	*true*	*true*
false	*true*	*false*	*false*	*false*	*true*	*false*	*true*	*true*	*true*	*true*	*true*	*true*
false	*true*	*false*	*false*	*false*	*true*	*true*	*true*	*true*	*true*	*true*	*true*	*true*
false	*true*	*false*	*false*	*true*	*false*	*false*	*true*	*false*	*false*	*true*	*true*	*false*
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
true	*true*	*true*	*true*	*true*	*true*	*true*	*false*	*true*	*true*	*false*	*true*	*false*

Enumerate rows (different assignments to symbols),
if KB is true in row, check that α is too

Inference by enumeration [R, 7/38]

Depth-first enumeration of all models is sound and complete

```
function TT-ENTAILS?(KB, α) returns true or false
   inputs: KB, the knowledge base, a sentence in propositional logic
           α, the query, a sentence in propositional logic

   symbols ← a list of the proposition symbols in KB and α
   return TT-CHECK-ALL(KB, α, symbols, [])
```

```
function TT-CHECK-ALL(KB, α, symbols, model) returns true or false
   if EMPTY?(symbols) then
      if PL-TRUE?(KB, model) then return PL-TRUE?(α, model)
      else return true
   else do
      P ← FIRST(symbols); rest ← REST(symbols)
      return TT-CHECK-ALL(KB, α, rest, EXTEND(P, true, model)) and
                 TT-CHECK-ALL(KB, α, rest, EXTEND(P, false, model))
```

$O(2^n)$ for n symbols; problem is **co-NP-complete**

Logical equivalence [R 7/39]

Two sentences are logically equivalent iff true in same models:
$\alpha \equiv \beta$ if and only if $\alpha \models \beta$ and $\beta \models \alpha$

$$
\begin{array}{rcll}
(\alpha \land \beta) & \equiv & (\beta \land \alpha) & \text{commutativity of } \land \\
(\alpha \lor \beta) & \equiv & (\beta \lor \alpha) & \text{commutativity of } \lor \\
((\alpha \land \beta) \land \gamma) & \equiv & (\alpha \land (\beta \land \gamma)) & \text{associativity of } \land \\
((\alpha \lor \beta) \lor \gamma) & \equiv & (\alpha \lor (\beta \lor \gamma)) & \text{associativity of } \lor \\
\neg(\neg\alpha) & \equiv & \alpha & \text{double-negation elimination} \\
(\alpha \Rightarrow \beta) & \equiv & (\neg\beta \Rightarrow \neg\alpha) & \text{contraposition} \\
(\alpha \Rightarrow \beta) & \equiv & (\neg\alpha \lor \beta) & \text{implication elimination} \\
(\alpha \Leftrightarrow \beta) & \equiv & ((\alpha \Rightarrow \beta) \land (\beta \Rightarrow \alpha)) & \text{biconditional elimination} \\
\neg(\alpha \land \beta) & \equiv & (\neg\alpha \lor \neg\beta) & \text{De Morgan} \\
\neg(\alpha \lor \beta) & \equiv & (\neg\alpha \land \neg\beta) & \text{De Morgan} \\
(\alpha \land (\beta \lor \gamma)) & \equiv & ((\alpha \land \beta) \lor (\alpha \land \gamma)) & \text{distributivity of } \land \text{ over } \lor \\
(\alpha \lor (\beta \land \gamma)) & \equiv & ((\alpha \lor \beta) \land (\alpha \lor \gamma)) & \text{distributivity of } \lor \text{ over } \land
\end{array}
$$

- The lecture continues (until its final form is completed) with the slides from [R ch. 7] starting with slide 40

BIBLIOGRAPHY

- [RN] Russel S., Norvig P. – Artificial Intelligence – A Modern Approach, 2nd ed. Prentice Hall, 2003 (1112 pages)
- [R] Stuart Russel – Course slides (visited oct. 2012 at http://aima.cs.berkeley.edu/instructors.html#homework)
- [W1] Mark Watson – Practical Artificial Intelligence Programming With Java AI 3rd ed., 2008
- [C] D. Cârstoiu – Sisteme Expert, Editura ALL, București, 1994