Logical agents

- Reading: Russell and Norvig, ch. 7. These notes are an abbreviated version of http://aima.eecs.berkeley.edu/slides-pdf/chapter07.pdf
- Knowledge-based agents
- Logic
- Propositional logic
- Forward and backward chaining

Where we are

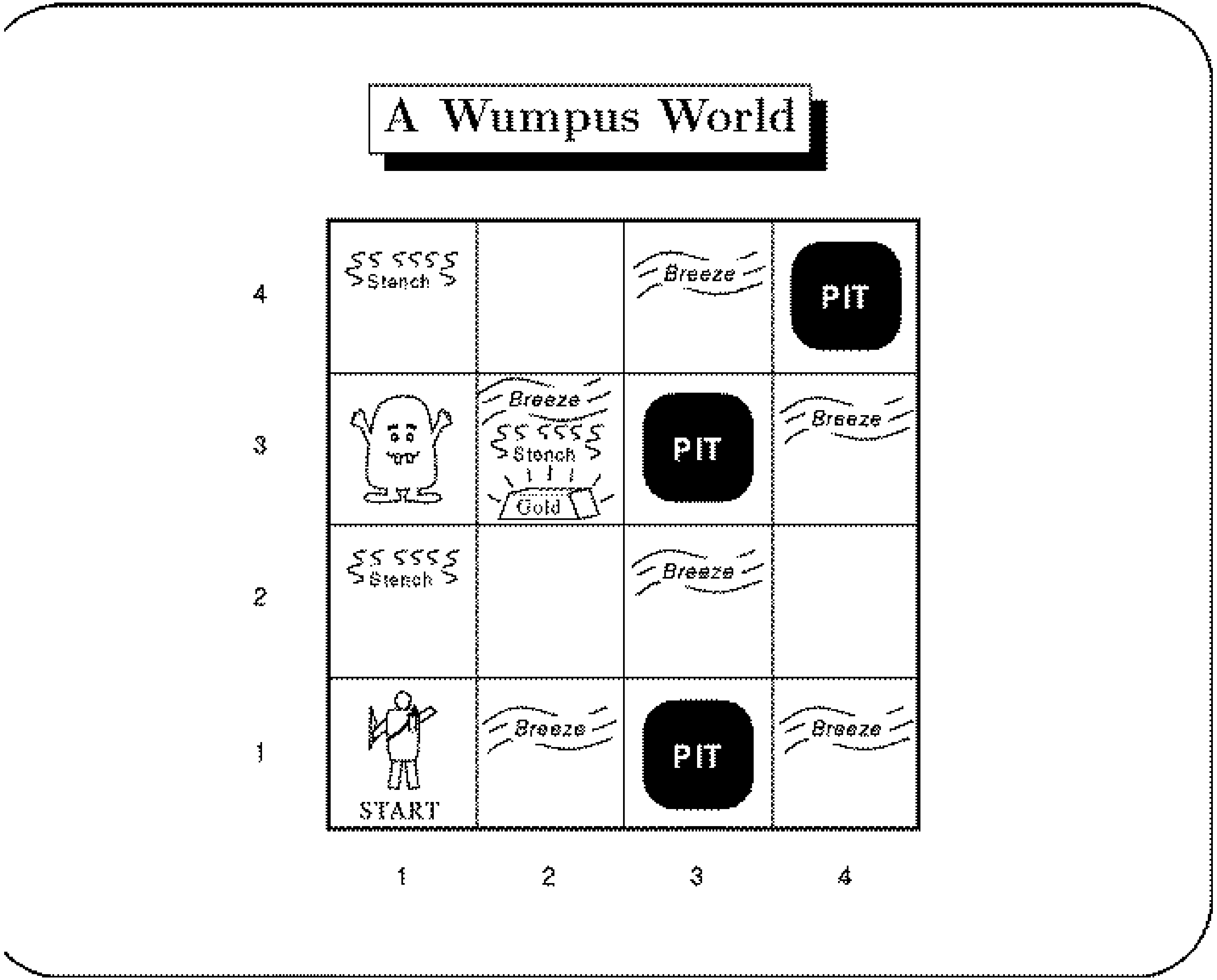
- Agents are given or collect information
- Information is sufficient to determine if goal state has been reached
- Data structure/database sufficient to store representation of “knowledge”

Knowledge-based agents

- Knowledge base = “set of sentences in a formal language”
    - Like the facts in a police investigation
    - formal language supports
        - Input of facts from any domain – what you “tell” the agent
        - use of *inference engine* to generate new facts and choose actions based on existing ones
        - Inference engine is domain-<u>independent</u>
        - Like detectives make deductions from the facts in an investigation
        - Agent can “ask” itself what to do
- Adds logic (reasoning) to agent

Wumpus

- Goal: get gold
- Score: gold +1000 death -1000 -1 per step -10 for using arrow
- Environment:
    - Squares next to Wumpus are smelly
    - Squares next to pits are breezy
    - Glitter if gold is in same square
    - Shooting arrow kills Wumpus if facing it
    - Only 1 arrow

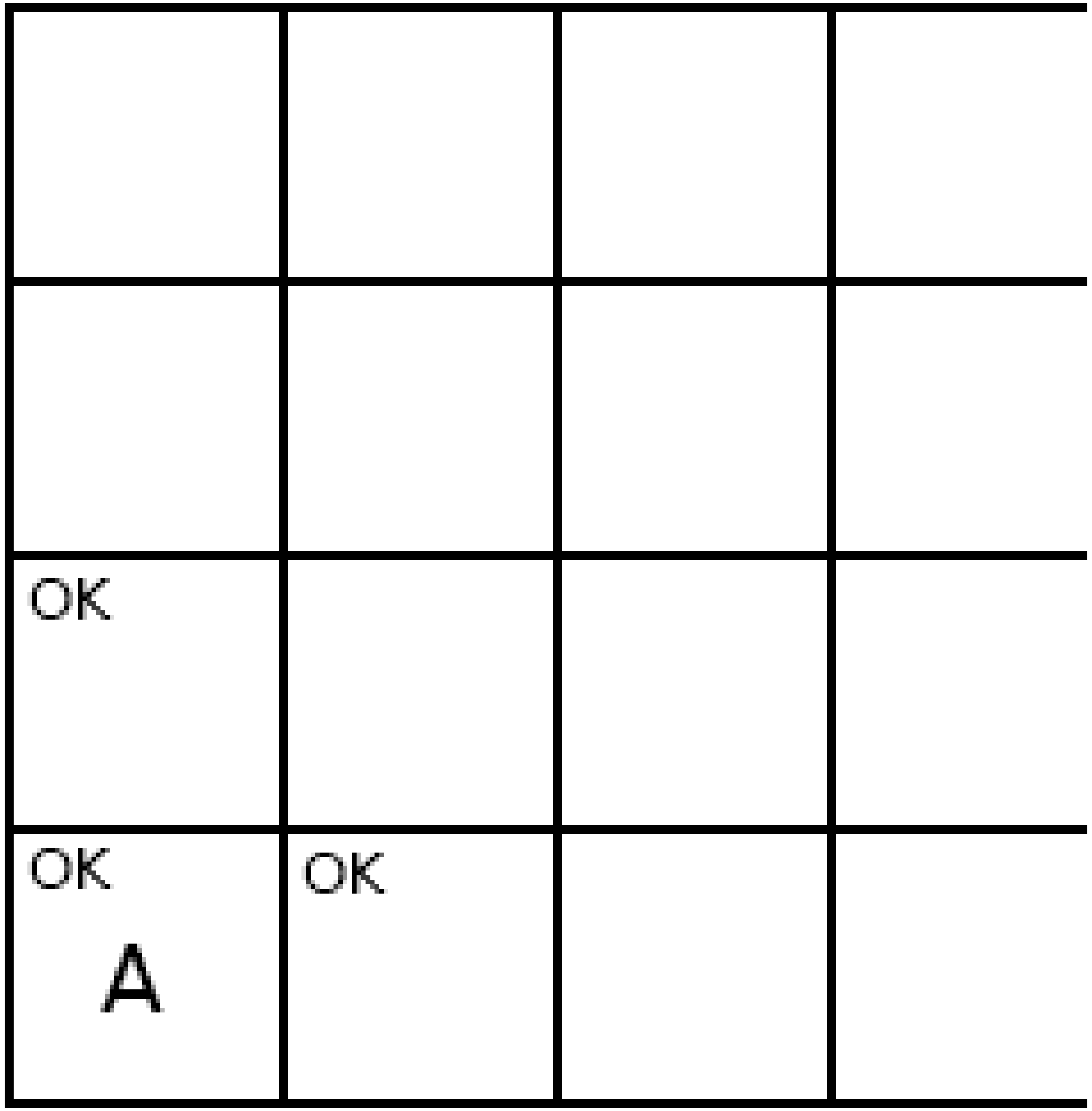


Wumpus world comments

- (i, j) = ith column, jth row
- Wumpus does not move
  - Configuration of game does change each time
- Moving onto a square with the Wumpus or a pit results in death

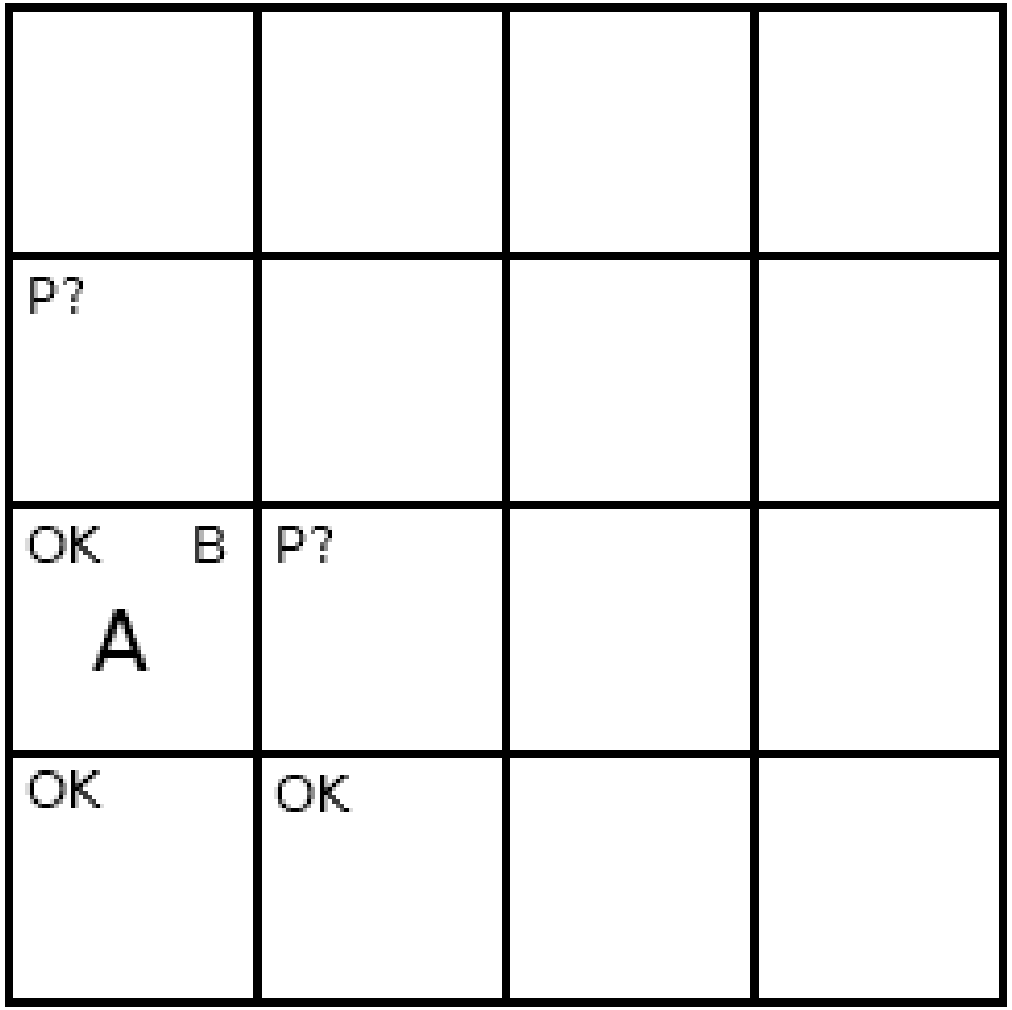
Reasoning in Wumpus world

- At start, no alerts (B = breezy, S = smelly), so squares above and to the right are safe



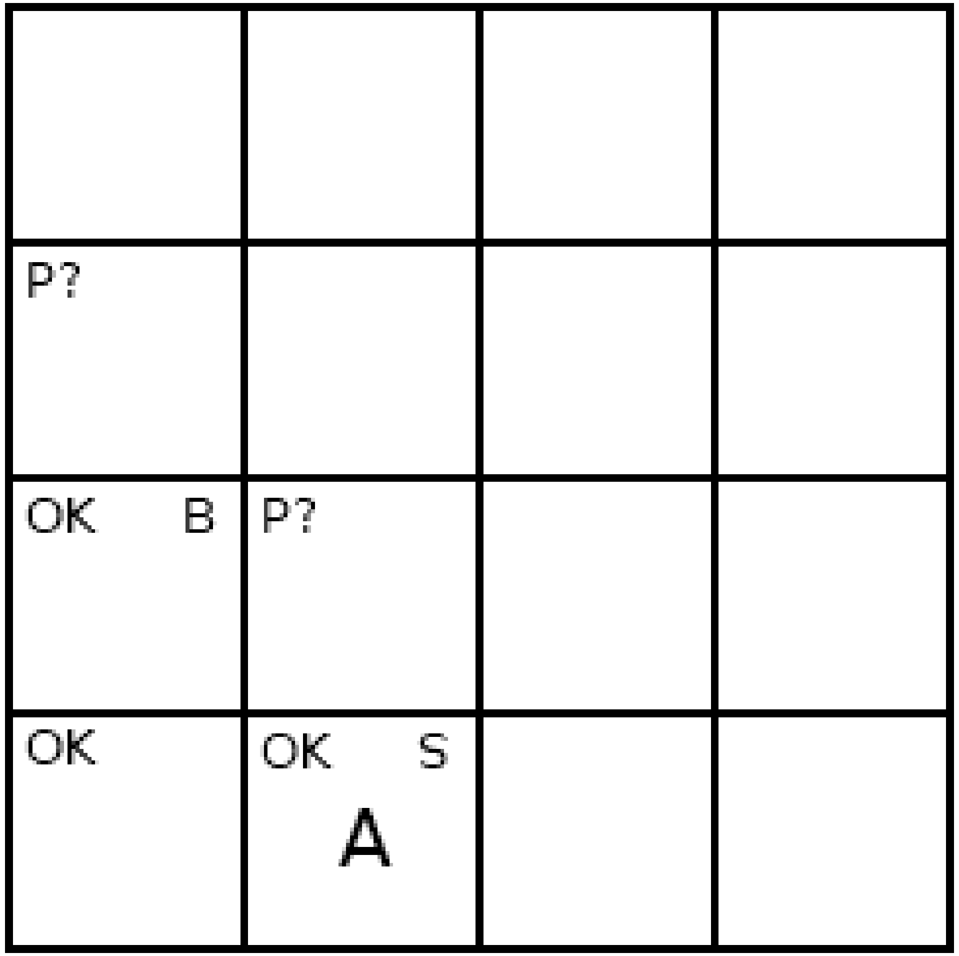
Reasoning in Wumpus world

- If agent moves up, senses Breezy
- Pit is either above or to the right



Reasoning in Wumpus world

- If agent tries to the right of initial square, Smelly alert goes off, but not Breezy
- What does this say about the square at (2,2)? (3,1) [3rd square in bottom row]? (1,3)?



Logic in general

- A *logic* is a formal language for representing information that allows conclusions to be made
- Syntax defines form of a *sentence* in the language
- Semantics defines meaning
  - For a logic, this defines whether a sentence is true or false
- KB $\models \alpha$ (a sentence)
  - KB *entails* α: if KB is true, α is true

Inference

- KB $\vdash_i \alpha$ = sentence α can be derived from KB by procedure *i*
- Soundness:
  - *i* is sound if whenever KB $\vdash_i \alpha$ it is also true that KB $\models \alpha$
- Completeness:
  - *i* is complete if whenever KB $\models \alpha$ it is also true that KB $\vdash_i \alpha$
- Goal here is to have inference procedure to allow us to make conclusions from the knowledge we have (KB)

Propositional logic

- Simplest logic
- Proposition symbols P_1, P_2 are sentences
  - Wumpus: B, G, S at each square are examples
- If S is a sentence, $\neg$ S is a sentence (negation)
- If S_1 and S_2 are sentences, $S_1 \wedge S_2$ is a sentence (conjunction)
- If S_1 and S_2 are sentences, $S_1 \vee S_2$ is a sentence (disjunction)
- If S_1 and S_2 are sentences, $S_1 \Rightarrow S_2$ is a sentence (implication)
- If S_1 and S_2 are sentences, $S_1 \Leftrightarrow S_2$ is a sentence (biconditional)

Propositional logic semantics

- Rules for evaluating truth with respect to a model *m*
  - S is true iff $\neg$ 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
  - $S_1 \Leftrightarrow S_2$ is true iff $S_1 \Rightarrow S_2$ is true and $S_2 \Rightarrow S_1$ is true
    - Equivalent to $S_1 = S_2$

Logical equivalence

- $\wedge$, $\vee$ are commutative and associative, can distribute one over the other
- $\alpha \equiv \neg(\neg\alpha)$
- $\alpha \Rightarrow \beta \equiv \neg\beta \Rightarrow \neg\alpha$ (contraposition)
- $\alpha \Rightarrow \beta \equiv \neg\alpha \vee \beta$ (implication elimination)
- $\alpha \Leftrightarrow \beta \equiv \alpha \Rightarrow \beta \wedge \beta \Rightarrow \alpha$
- $\neg(\alpha \wedge \beta) \equiv \neg\alpha \vee \neg\beta$ (De Morgan)
- $\neg(\alpha \vee \beta) \equiv \neg\alpha \wedge \neg\beta$ (De Morgan)

Validity and satisfiability

- A sentence is *valid* iff it is true in all models
  - Ex: $A \vee \neg A$
  - $KB \models \alpha$ iff $(KB \Rightarrow \alpha)$ is valid
- A sentence is *satisfiable* iff it is true in some model
- A sentence is *unsatisfiable* iff it is not true in any model
  - $KB \models \alpha$ iff $(KB \wedge \neg\alpha)$ is unsatisfiable (proof by contradiction)

Wumpus example

- Let $P_{i,j}$ be true iff there is a pit in [i, j]
- Let $B_{i,j}$ be true iff there is a breeze in [i, j]
- In sample world, $\neg P_{1,1}$, $\neg B_{1,1}$, $B_{2,1}$
  - $B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$
  - $B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$

Forward chaining

- From starting propositions, use inference rules to generate more sentences to store in the KB
- If trying to determine if a goal sentence is true, may waste a lot of time generating new sentences that do not help lead to goal sentence

Backward chaining

- Work backward from query q
  - Check if q is known in KB already
    - if true, done
    - If not, use backward chaining to prove all premises of q
- To avoid loops, check if subgoal is already on goal stack
- To avoid repeated work, check if new subgoal has already been proved true or has already failed
- Makes search much more efficient