Lecture 7: Propositional Logic

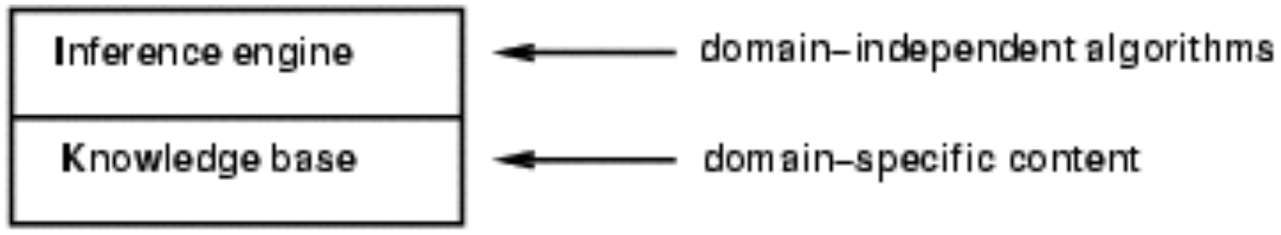
Artificial Intelligence
CS-UY-4613-A / CS-GY-6613-I

Julian Togelius
julian.togelius@nyu.edu

Outline

- Knowledge-based agents
- The Wumpus world
- Logic: models and entailment
- Propositional logic
- Equivalence, validity, satisfiability
- Model checking
- Inference

Knowledge bases



- Knowledge base = set of sentences in a formal language

- Declarative approach; the procedural part is in the domain-general inference engine

- “Tell”: add sentences to the KB

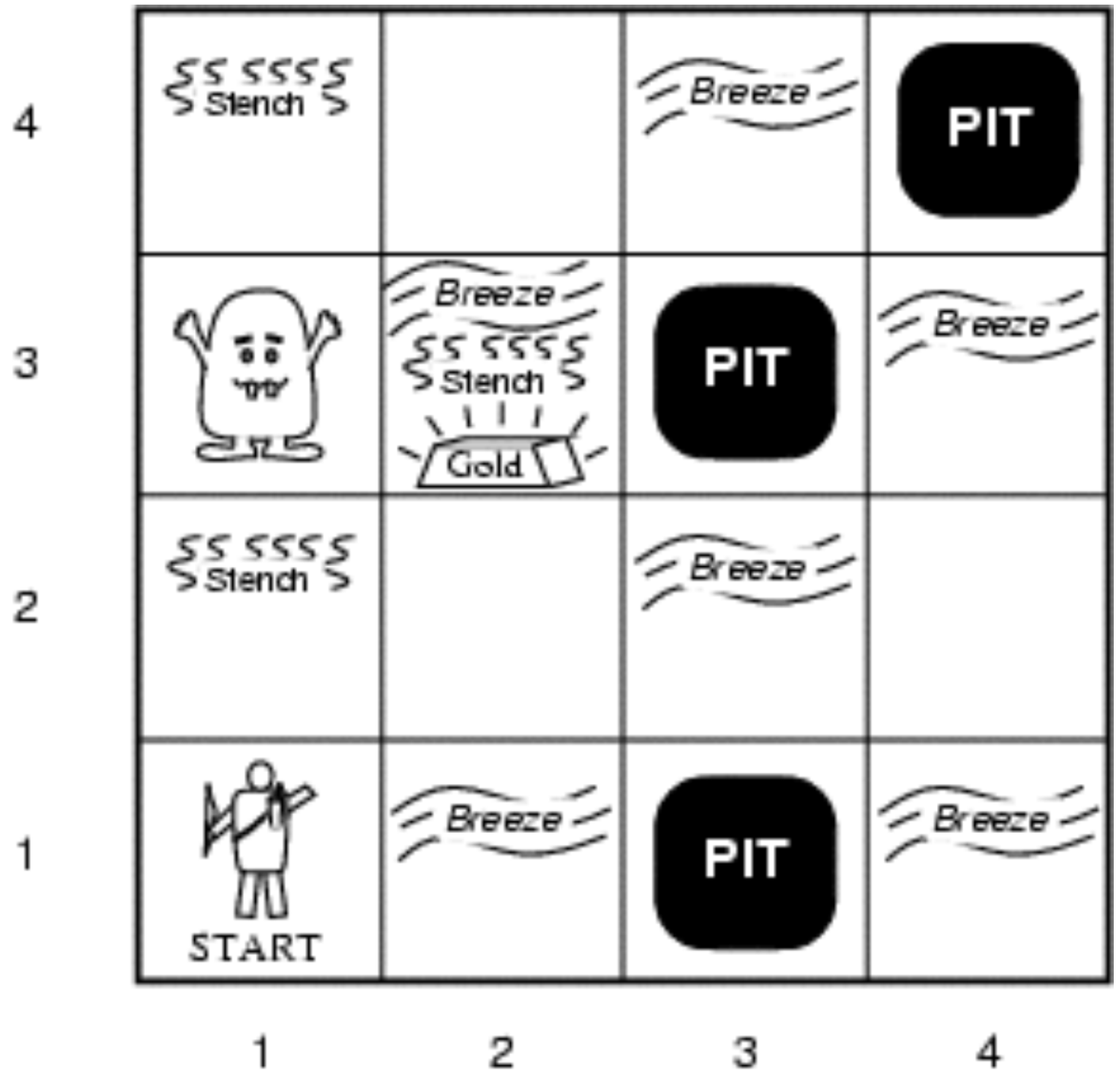
- “Ask”: get answers needed for action

Knowledge-based 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*))
  $t \leftarrow t + 1$
  return *action*

Hunt the Wumpus



- **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

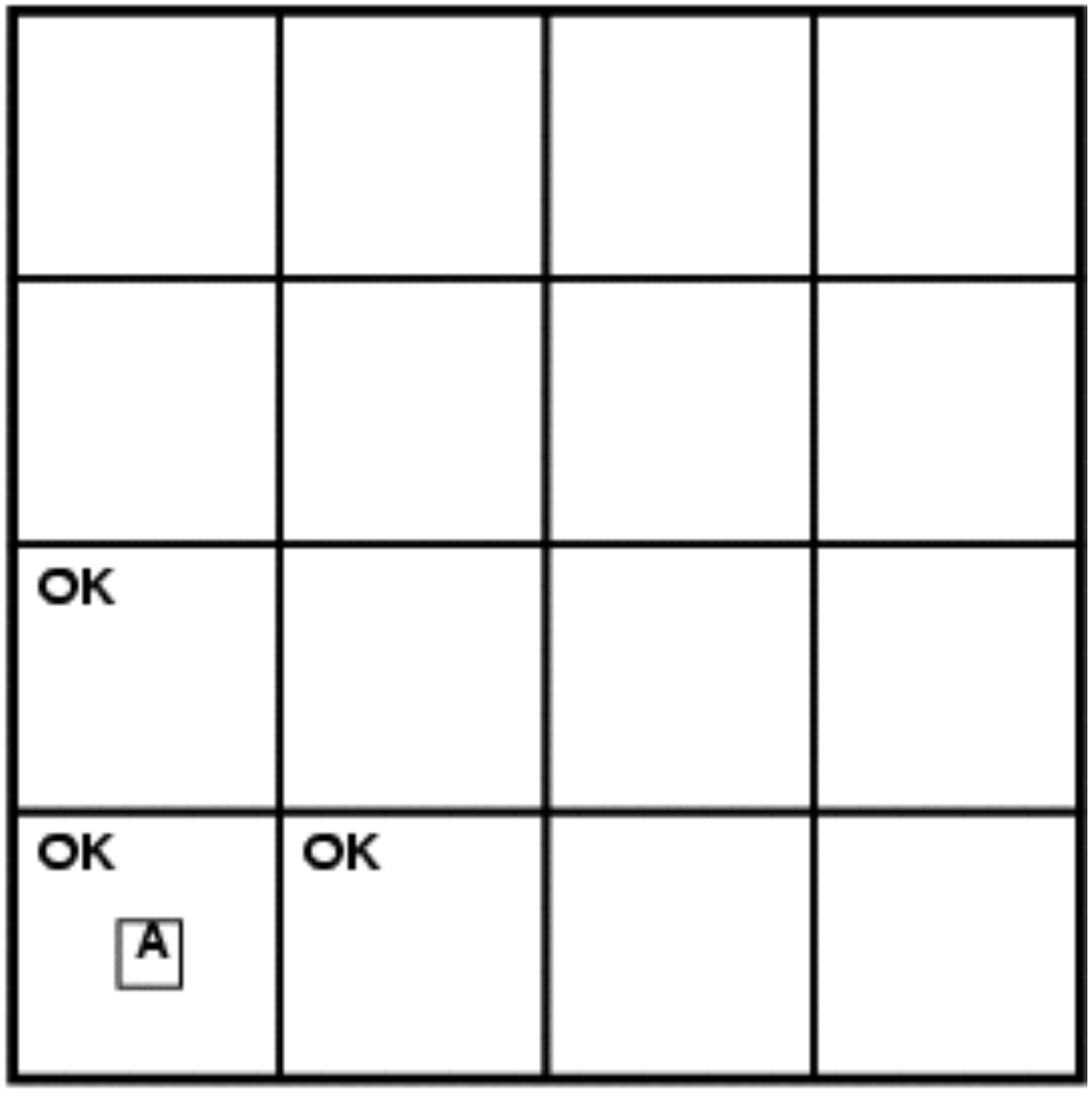
- **Sensors:** Stench, Breeze, Glitter, Bump, Scream

- **Actuators:** Left turn, Right turn, Forward, Grab, Release, Shoot

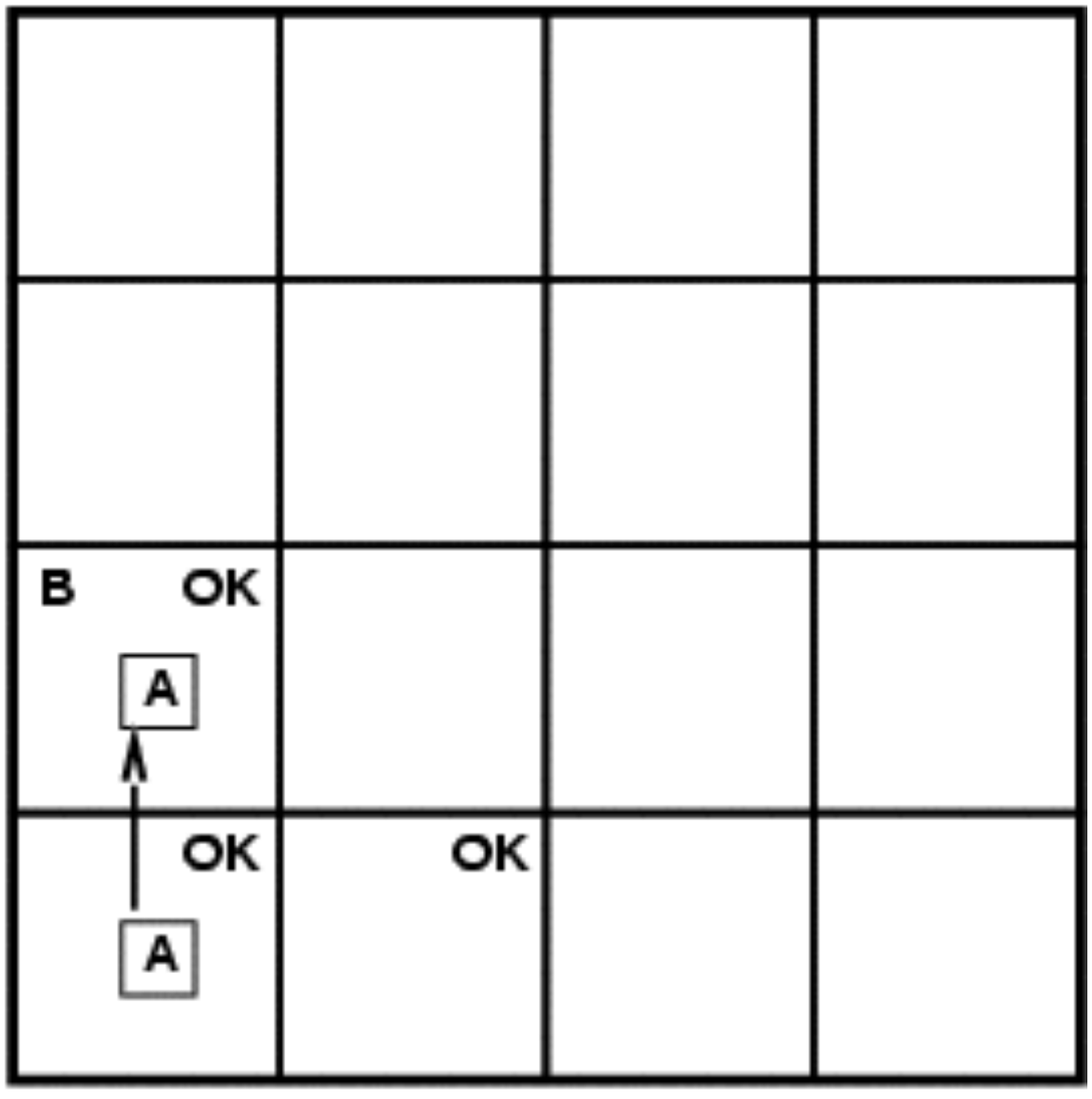
Hunt the Wumpus

- *Fully Observable?* No – only local perception
- *Deterministic?* Yes
- *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

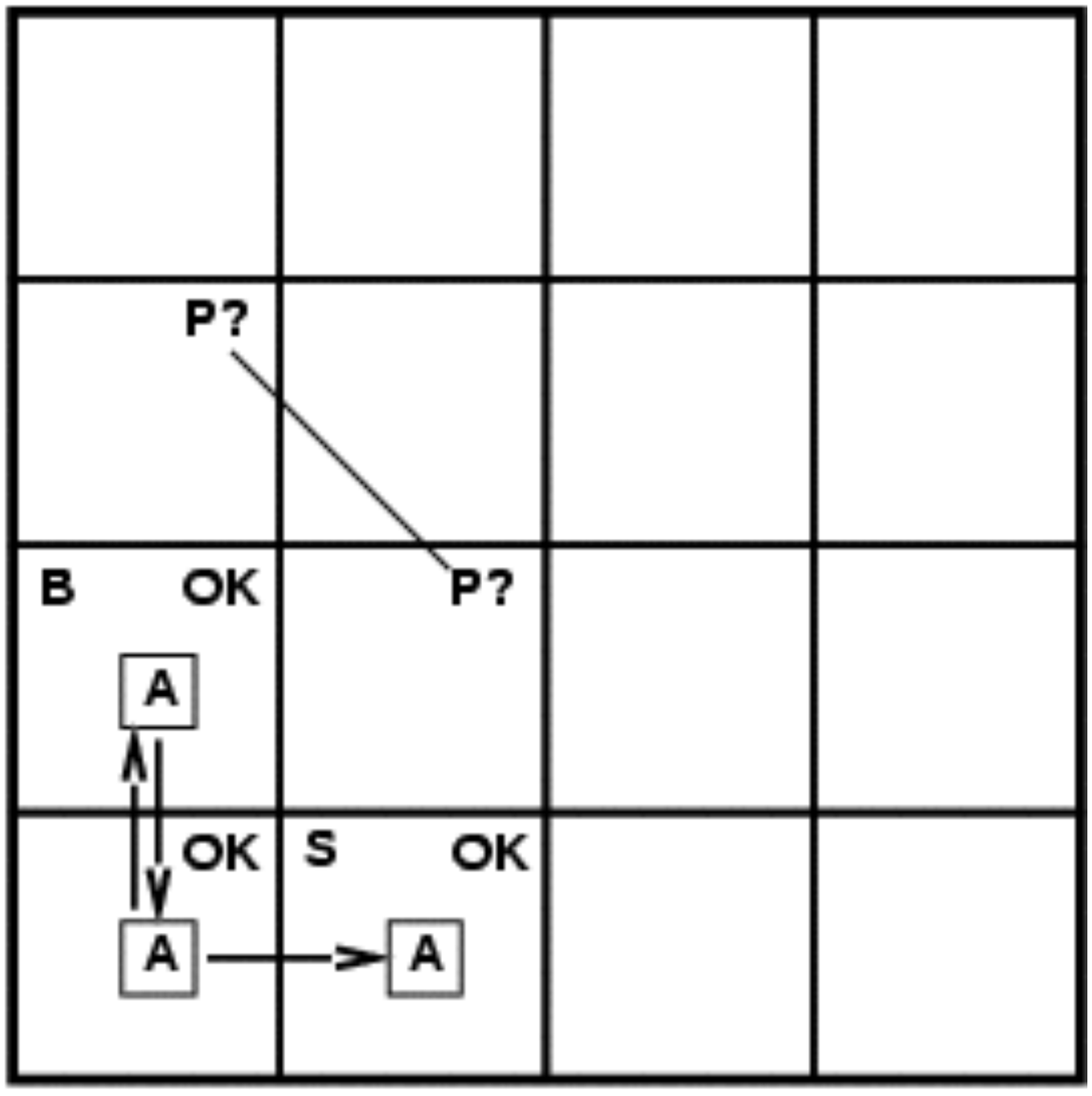
Exploring a Wumpus world



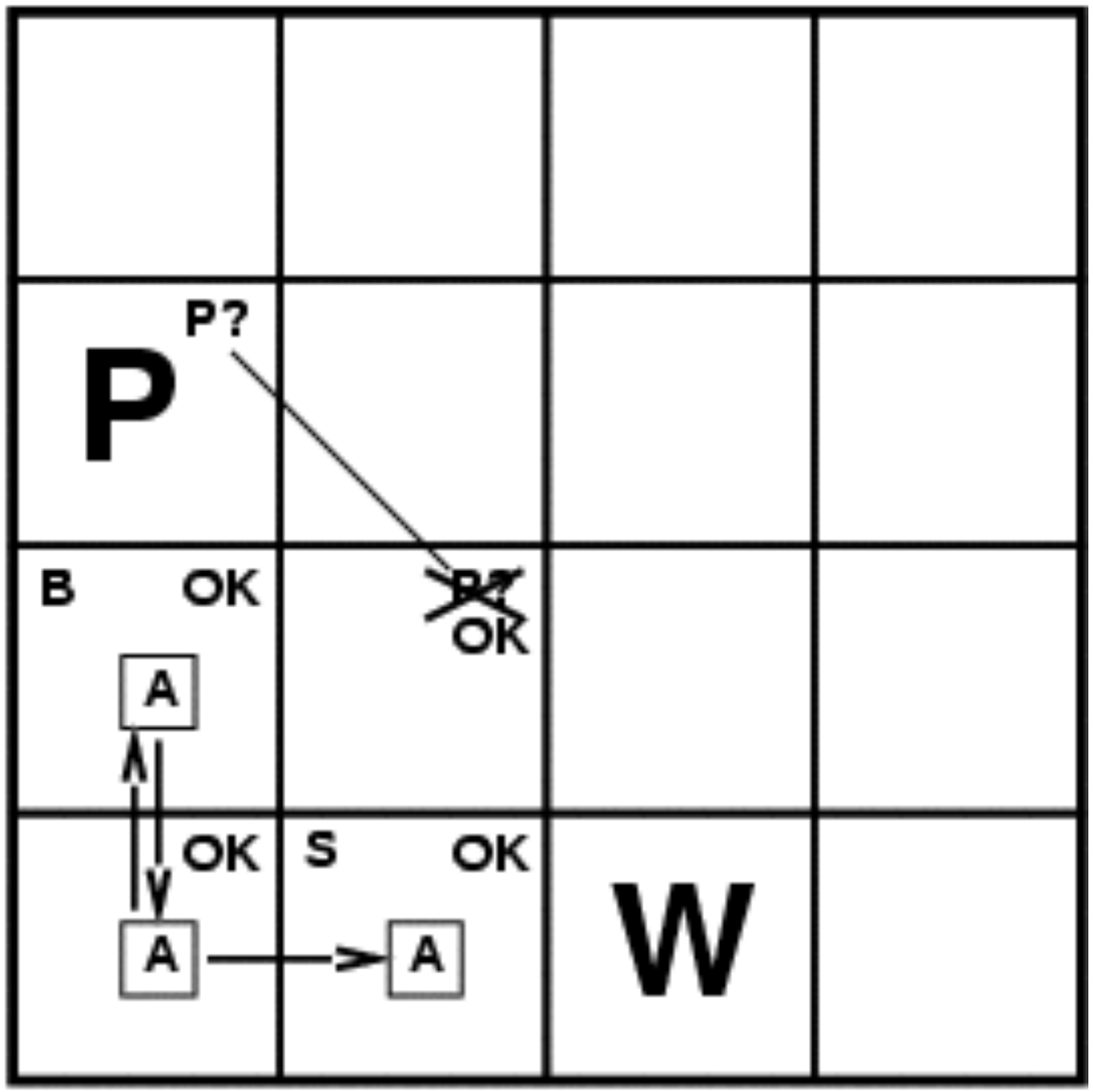
Exploring a Wumpus world



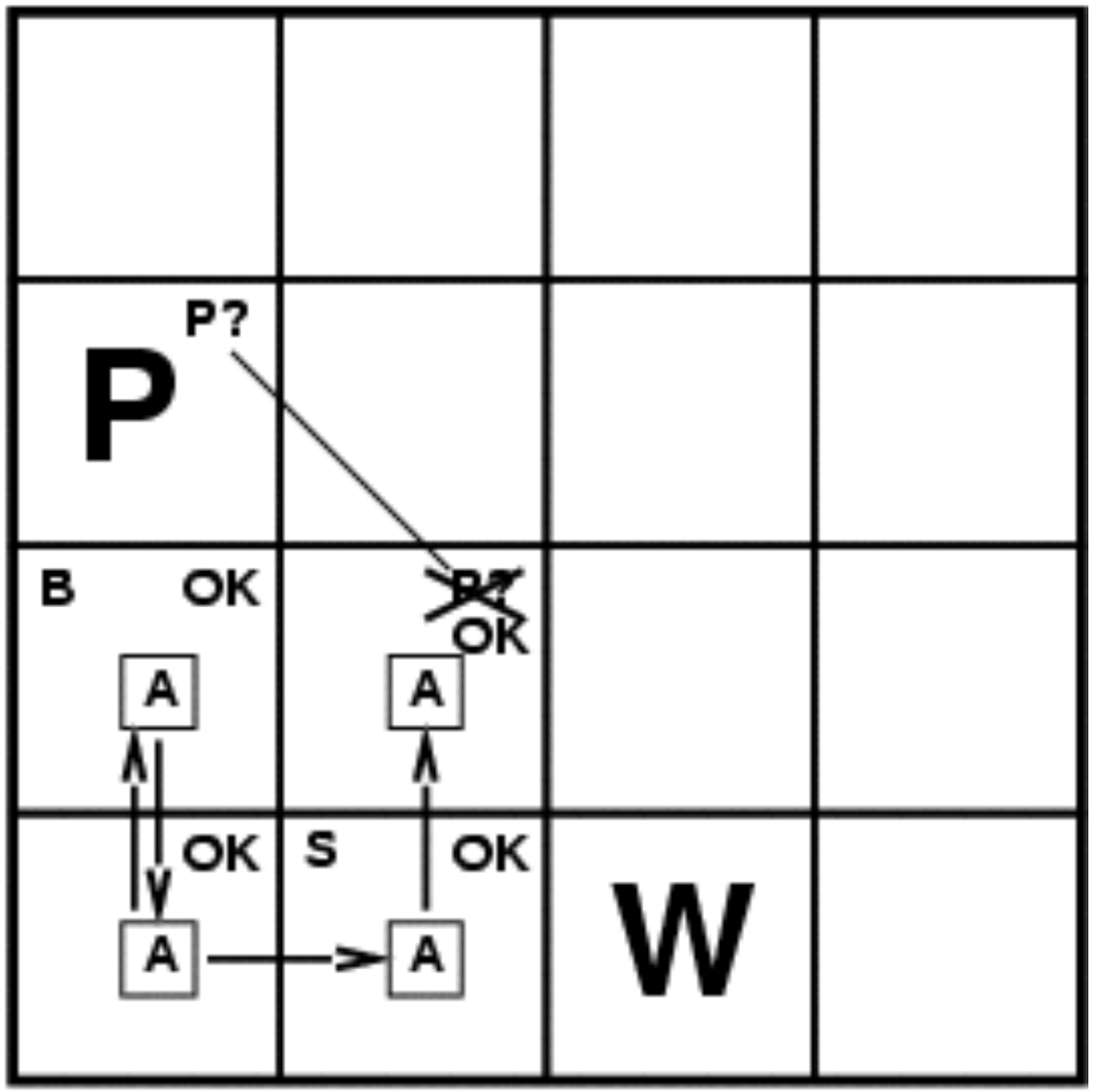
Exploring a Wumpus world



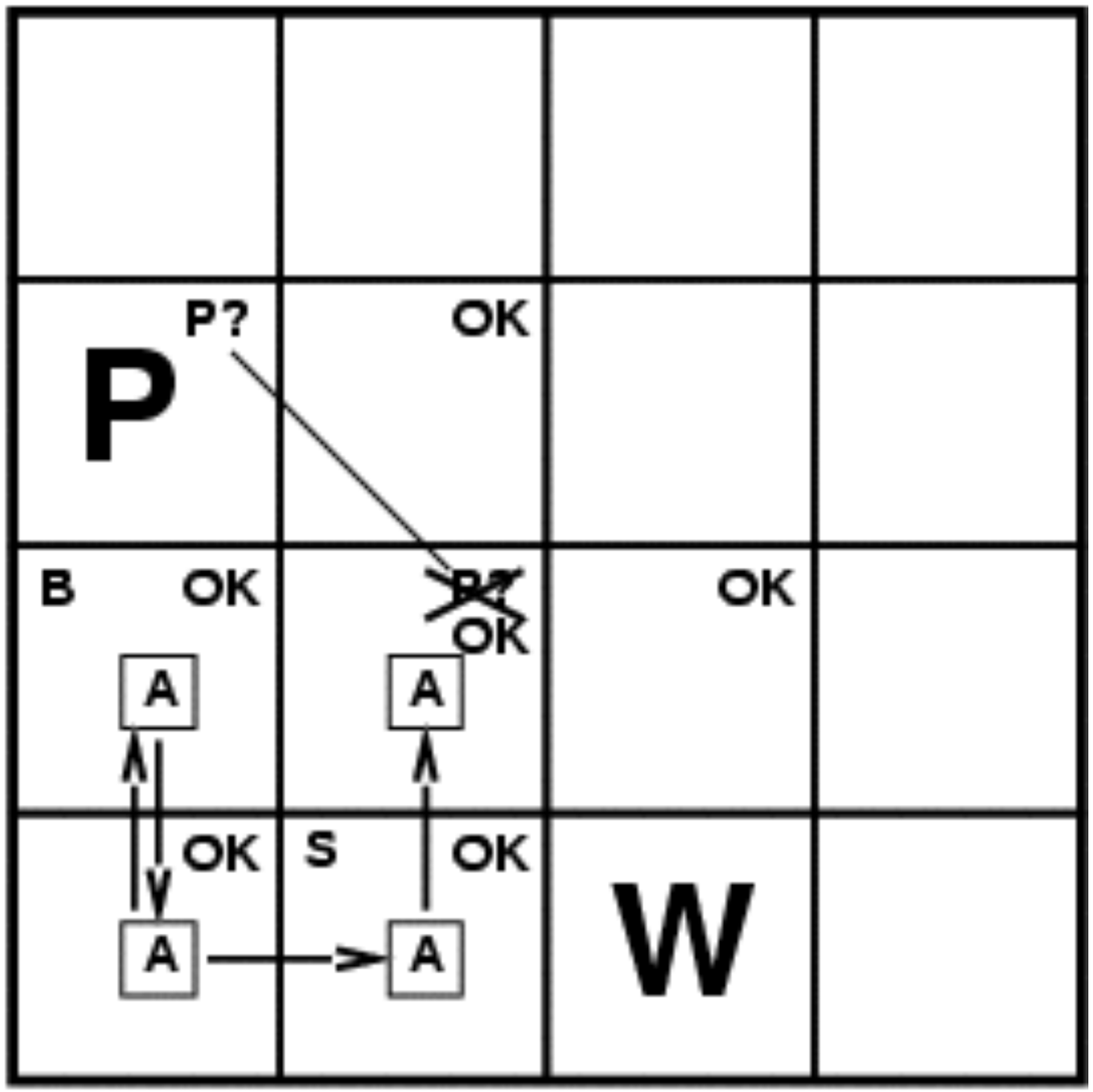
Exploring a Wumpus world



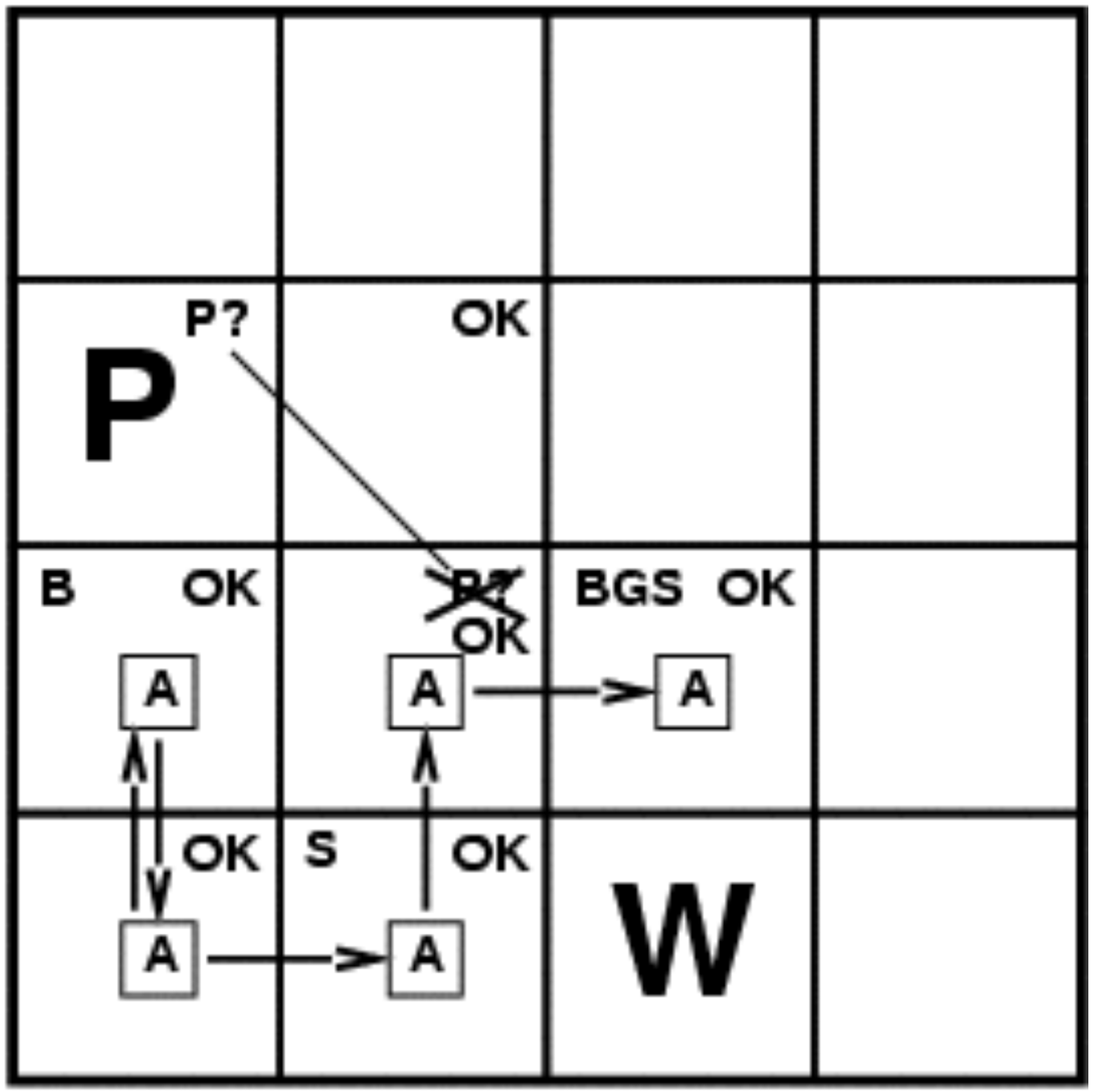
Exploring a Wumpus world



Exploring a Wumpus world



Exploring a Wumpus world



Logic

- *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

A logic: arithmetic

- x+2 ≥ y is a sentence; x2+y > {} is not a sentence
- x+2 ≥ y is true iff the number x+2 is no less than the number y
- x+2 ≥ y is true in a world where x = 7, y = 1
- x+2 ≥ y is false in a world where x = 0, y = 6

Entailment

- *Entailment* means that one thing follows from another: KB $\models$ α

- Knowledge base KB entails sentence α if and only if α is true in all worlds where KB is true

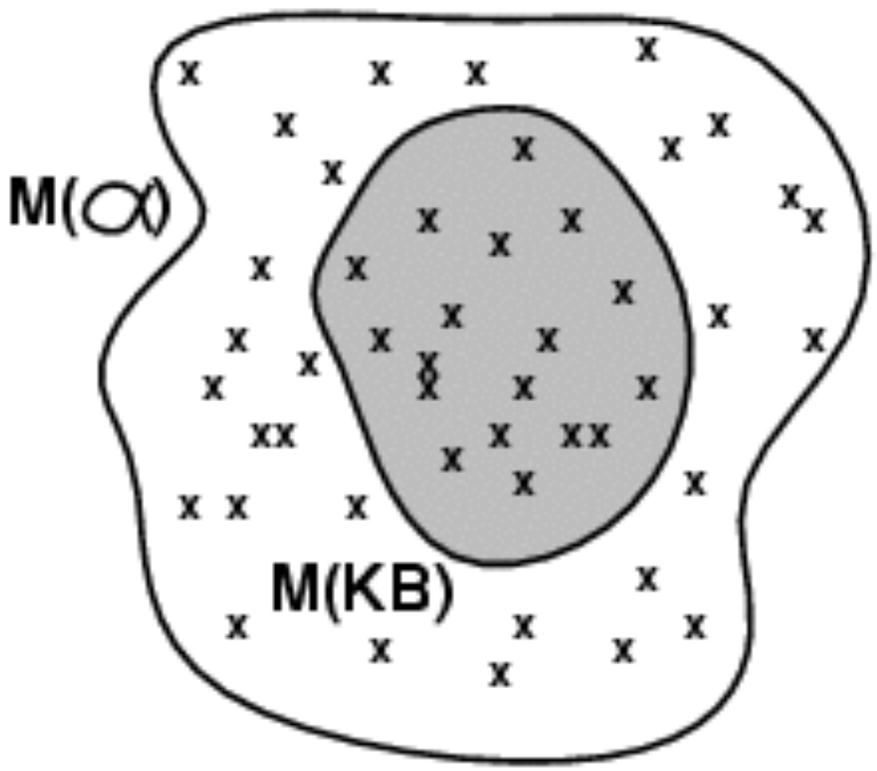
    - E.g., the KB containing “Red team won” and “Blue team won” entails “Either Red or Blue team won”

    - E.g., x+y = 4 entails 4 = x+y

- Entailment is a relationship between sentences (i.e. *syntax*) that is based on *semantics*

Models

- 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(α) is the set of all models of α
- Then KB $\models$ α iff M(KB) ⊆ M(α)
- E.g. KB = Blue won and Red won
  α = Blue won

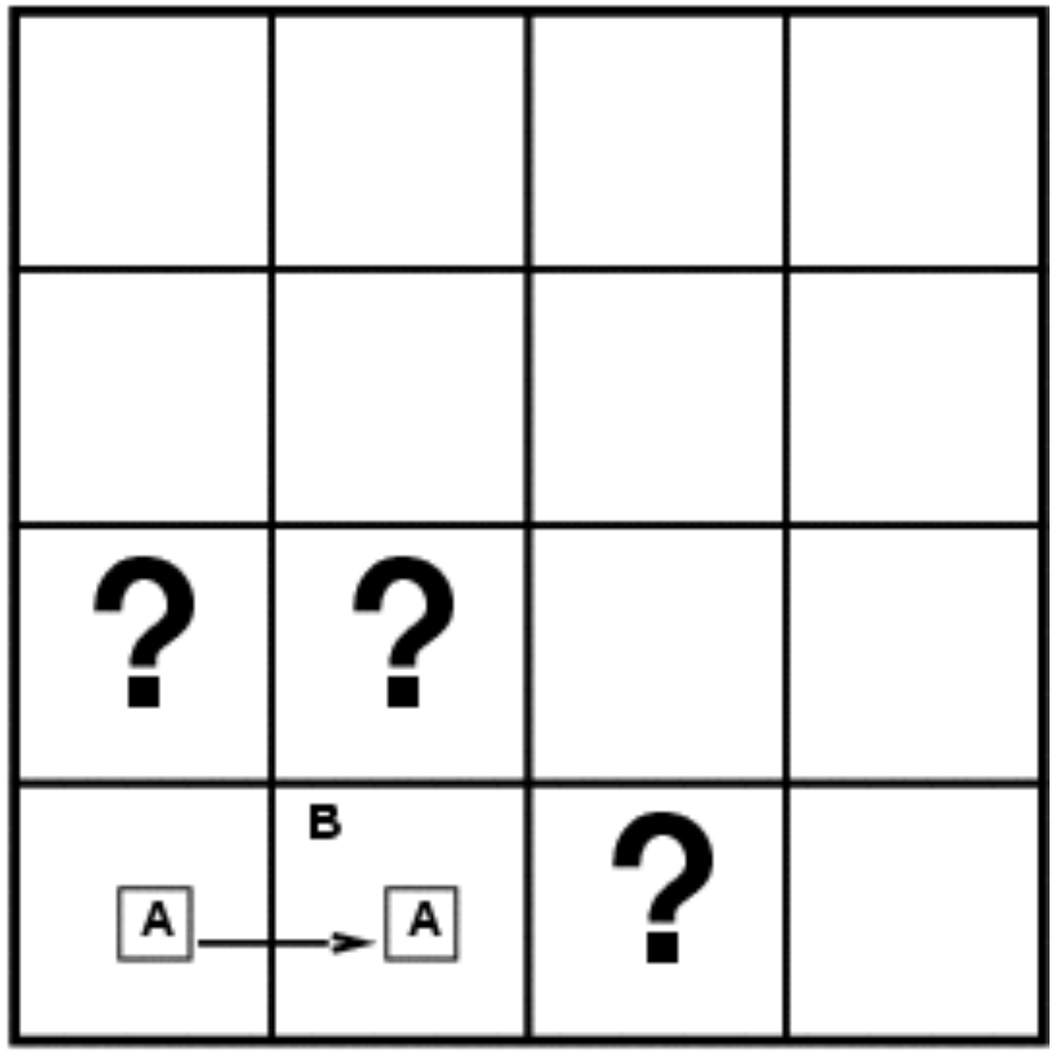


Entailment in Wumpus

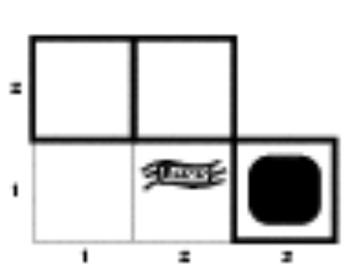
Situation after detecting nothing in [1,1], moving right, breeze in [2,1]

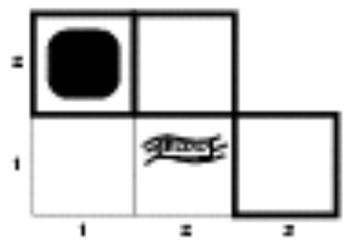
Consider possible models for *KB* assuming only pits

3 Boolean choices ⇒ 8 possible models

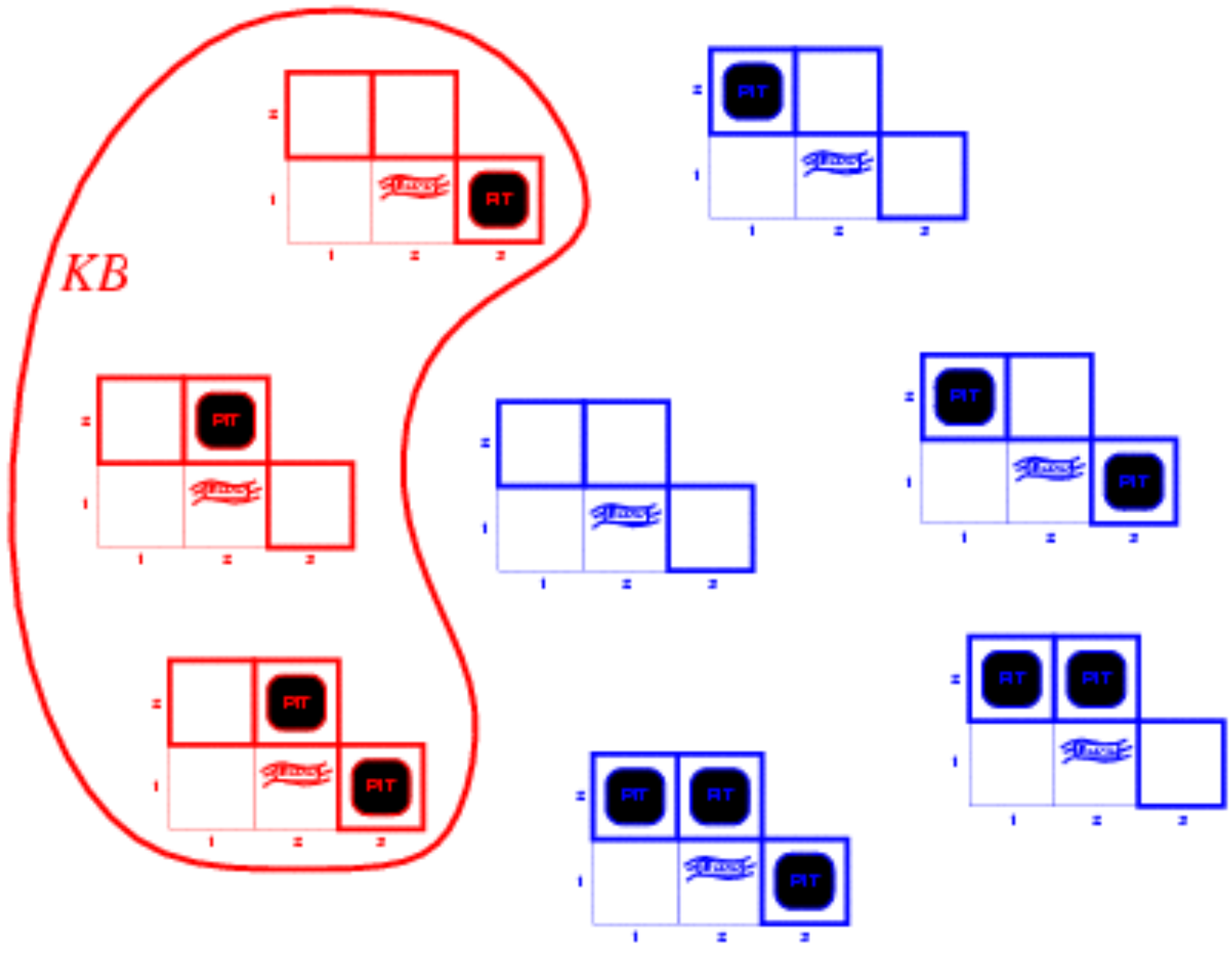


Wumpus models



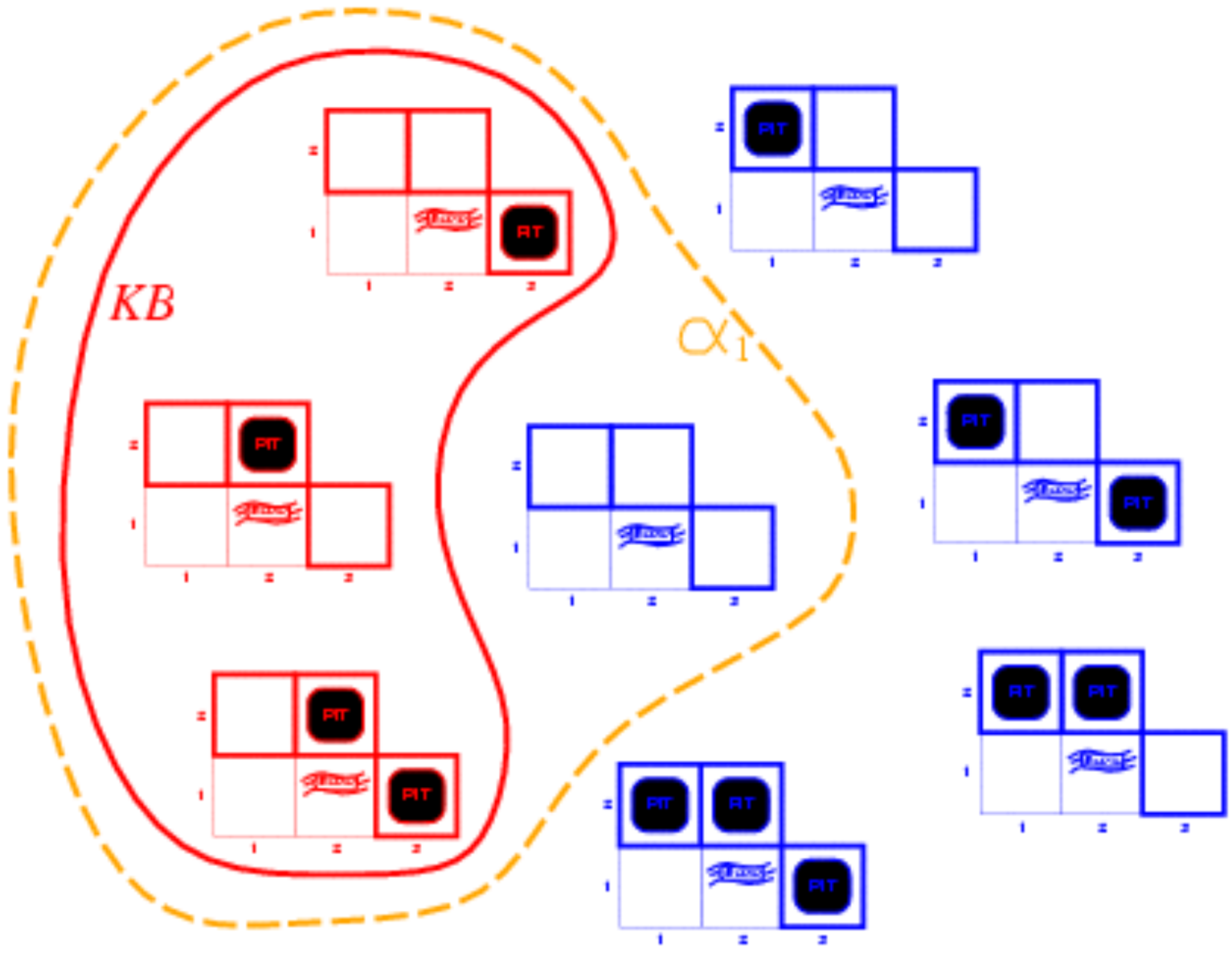


Wumpus models



- *KB* = wumpus-world rules + observations

Wumpus models



α_1 = "[1,2] is safe", KB $\models \alpha_1$, proved by model checking

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 $\vdash_i \alpha$
- Preview: we will define a logic (first-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

- Propositional logic is the simplest logic – 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 ∨ 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

Each model specifies true/false for each proposition symbol

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

» $\neg P_{1,2} \wedge (P_{2,2} \vee P_{3,1}) = \text{true} \wedge (\text{true} \vee \text{false}) = \text{true} \wedge \text{true} = \text{true}$

Truth table for connectives

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$	$P \Leftrightarrow Q$
$false$	$false$	$true$	$false$	$false$	$true$	$true$
$false$	$true$	$true$	$false$	$true$	$true$	$false$
$true$	$false$	$false$	$false$	$true$	$false$	$false$
$true$	$true$	$false$	$true$	$true$	$true$	$true$

Validity and satisfiability

- A sentence is valid if it is true in all models,
  e.g., True, $A \vee \neg A$, $A \Rightarrow A$, $(A \wedge (A \Rightarrow B)) \Rightarrow B$

- Validity is connected to inference via the Deduction Theorem:
  $KB \models \alpha$ if and only if $(KB \Rightarrow \alpha)$ is valid

- A sentence is satisfiable if it is true in some model
  e.g., $A \vee B$, C

- A sentence is unsatisfiable if it is true in no models
  e.g., $A \wedge \neg A$

- Satisfiability is connected to inference via the following:
  $KB \models \alpha$ if and only if $(KB \wedge \neg \alpha)$ is unsatisfiable

Two kinds of proof methods

- **Application of inference rules**
  - Legitimate (sound) generation of new sentences from old
  - Proof = a sequence of inference rule applications
    Can use inference rules as operators in a standard search algorithm
- **Model checking**
  - truth table enumeration (always exponential in *n*)
  - improved backtracking, e.g., Davis--Putnam-Logemann-Loveland (DPLL)
  - heuristic search in model space (sound but incomplete), e.g., min-conflicts-like hill-climbing algorithms

Model search: WalkSat

- Incomplete, local search algorithm
- Evaluation function: The min-conflict heuristic of minimizing the number of unsatisfied clauses
- Balance between greediness and randomness

WalkSat

function WALKSAT(*clauses*, *p*, *max-flips*) **returns** a satisfying model or *failure*

inputs: *clauses*, a set of clauses in propositional logic
p, the probability of choosing to do a "random walk" move
max-flips, number of flips allowed before giving up

model ← a random assignment of *true*/*false* to the symbols in *clauses*
for $i = 1$ **to** *max-flips* **do**
  if *model* satisfies *clauses* **then return** *model*
  clause ← a randomly selected clause from *clauses* that is false in *model*
  with probability *p* flip the value in *model* of a randomly selected symbol from *clause*
  else flip whichever symbol in *clause* maximizes the number of satisfied clauses
return *failure*

Expressiveness limit of propositional logics

- KB contains "physics" sentences for every single square
- For every time t and every location [x,y],
- $L_{x,y} \wedge FacingRight_t \wedge Forward_t \Rightarrow L_{x+1,y}$
- Rapid proliferation of clauses

About the midterm

- Literature: course slides and relevant chapters in the book,
- Focus on understanding of concepts and algorithms; minimum of details and nitpicking
- Answers in English and/or pseudocode
- Probably some multiple choice questions
- Write legibly
- Write short

Examples from last year's AI for games midterm

- 1. For what kind of games can alpha-beta search be used in its "pure" form (without modifications)? Answer with the characteristics of those games where the method can be used. Also, give at least two examples of such games.

- 5. Will an evolutionary algorithm always find the optimal solution for any problem? If yes, why, and if no, why not?

- 9. In Monte Carlo Tree Search, how does the tree expansion differ from Minimax? And what is typically used in place of a state evaluation function to determine the value of a node?

Project

- Not all algorithms need to get good results - explain in your report why

- Pac-Man: use a fast ghost controller, e.g. New Aggressive Ghosts

- Use the same controller for all of part 1

- Should be possible to input a time limit for ghosts - give them as little time as possible

- Keep 40ms limits on your moves

- A game state is a copy() of the Game object

Project

- You should write all the code that implements *the algorithms* yourself
- If you make use of any existing code, specify this very clearly in the project and source code comments
  - Point 1 still holds
- Methods that are called by the algorithms (e.g state evaluation functions, heuristic for A*) can use existing code, but make this clear

Project

- Implement your own Controller / Agent
- Do not change anything inside the game
- For adversarial games, choose a simple opponent to play against, which becomes part of your forward model. For example one step greedy search.
- For non-adversarial games, you don’t need to implement MiniMax

Extra office hours

- Room 872 in Magnet, 8th floor at 2 MetroTech
- Friday October 16, 1.30-2.30
- Tuesday October 20, 1.30-2.30