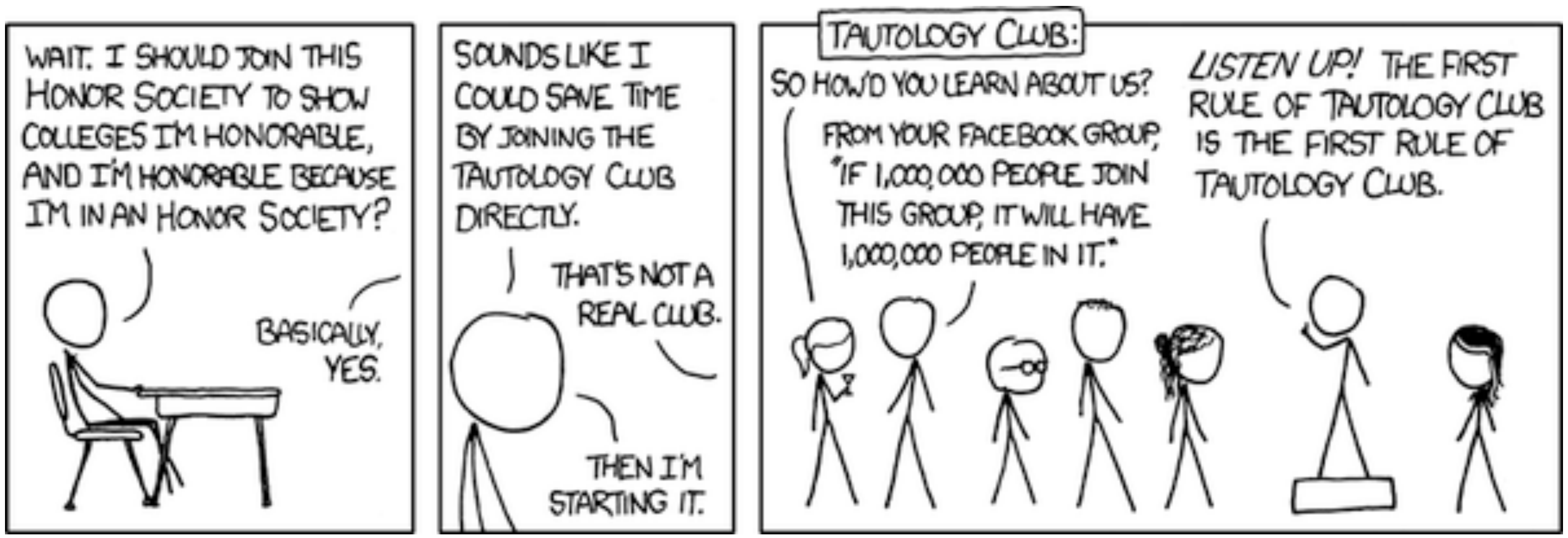
Announcements

- Summer Symposium on AI
  - July 25-26
  - https://sites.google.com/view/aisymposium2020/home
- Final exam
  - Aug. 26, 8am-9:40am

Ve492: Introduction to Artificial Intelligence

Logical Agent and Propositional Logic



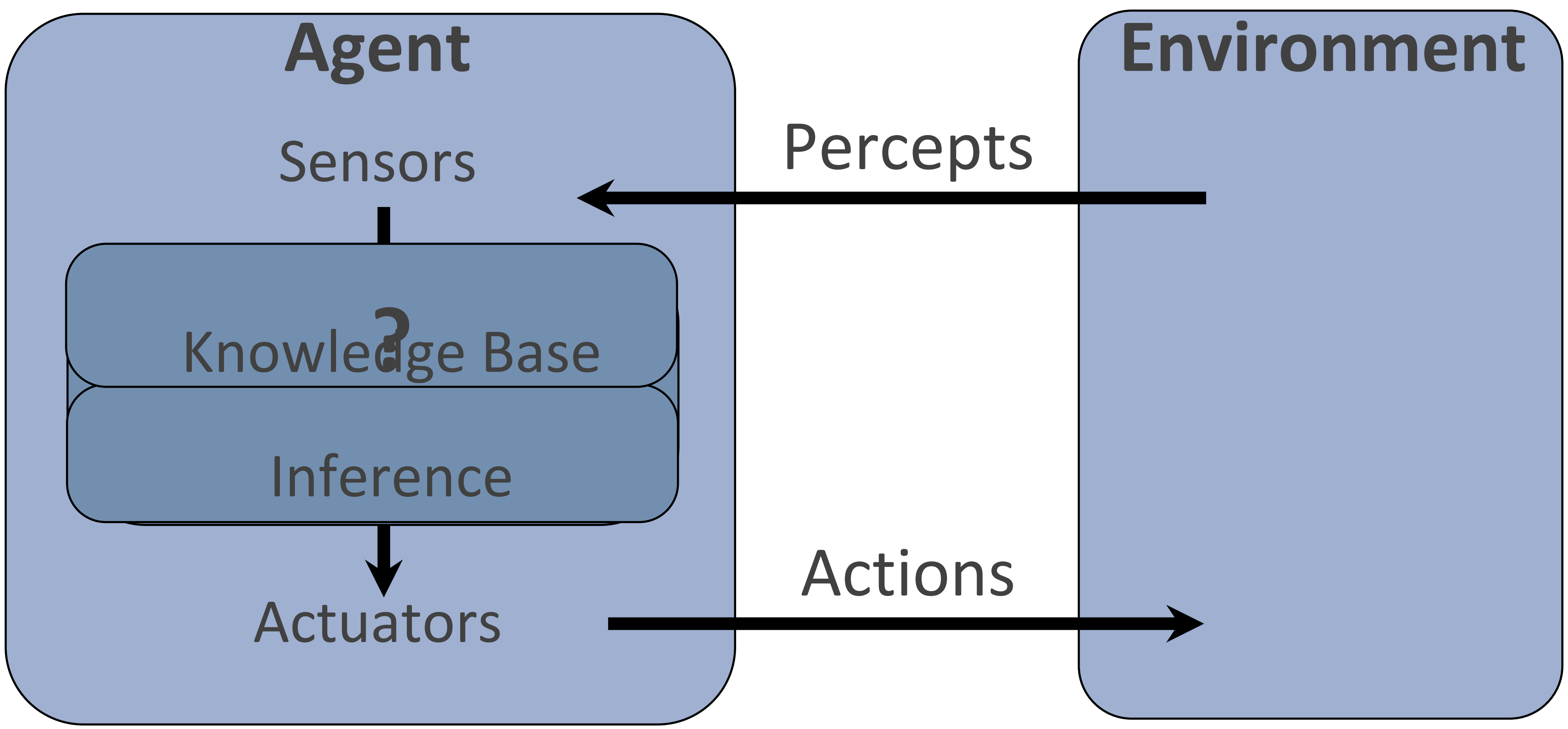
Paul Weng

UM-SJTU Joint Institute

Slides adapted from AIMA, UM, CMU

Logical Agents

Logical agents and environments



Wumpus World

Performance

- pick up gold = +1000,
- get eaten or fall in pit = -100
- step = -1
- shoot = -10

Environment

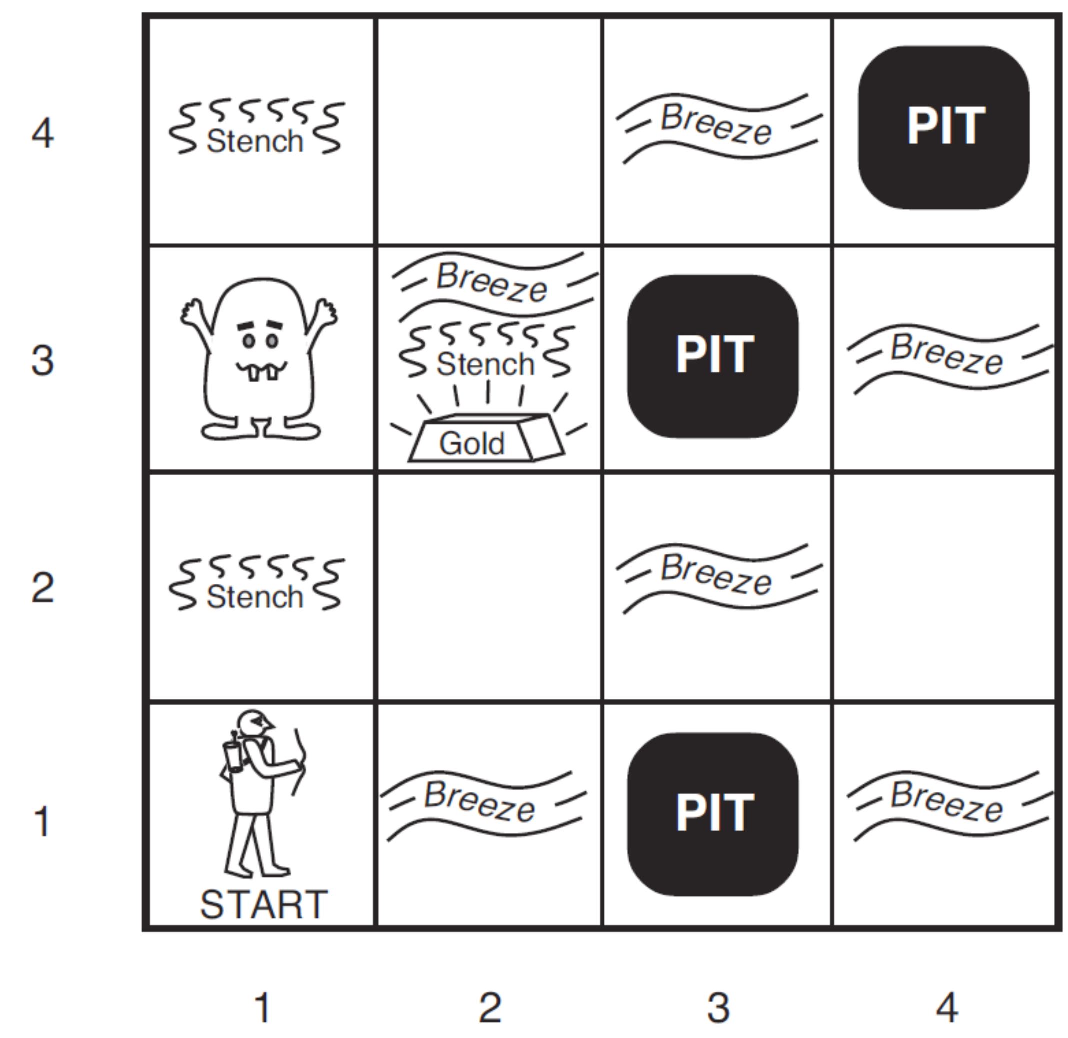
- grid

Actuators

- move forward,
- turn left or right,
- pick up,
- shoot

Sensors

- Stench,
- Breeze,
- Glitter,
- Bump,
- Scream



http://thiagodnf.github.io/wumpus-world-simulator/

A Knowledge-based Agent

```
function KB-AGENT(percept) returns an action
    persistent: KB, a knowledge base
                t, an integer, initially 0

    TELL(KB, PROCESS-PERCEPT(percept, t))

    action ← ASK(KB, PROCESS-QUERY(t))

    TELL(KB, PROCESS-RESULT(action, t))

    t←t+1

    return action
```

Logical Agents

So what do we TELL our knowledge base (KB)?

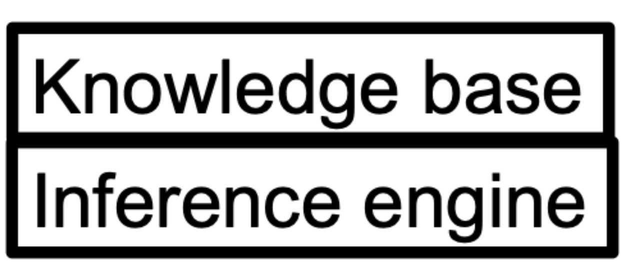
- Facts (sentences)
  - The grass is green
  - The sky is blue
- Rules (sentences)
  - Eating too much candy makes you sick
  - When you’re sick you don’t go to school
- Percepts and Actions (sentences)
  - Pat ate too much candy today

What happens when we ASK the agent?

- Inference – new sentences created from old
  - Pat is not going to school today

Knowledge

- Knowledge base = set of sentences in a formal language
- Declarative approach to building an agent (or other system):
- Tell it what it needs to know (or have it Learn the knowledge)
- Then it can Ask itself what to do—answers should follow from the KB
- Agents can be viewed at the knowledge level
  i.e., what they know, regardless of how implemented
- A single inference algorithm can answer any answerable question
  - Cf. a search algorithm answers only “how to get from A to B” questions

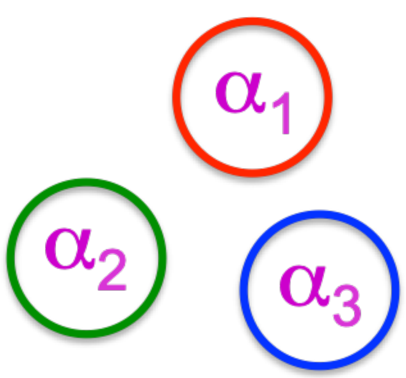


Domain-specific facts

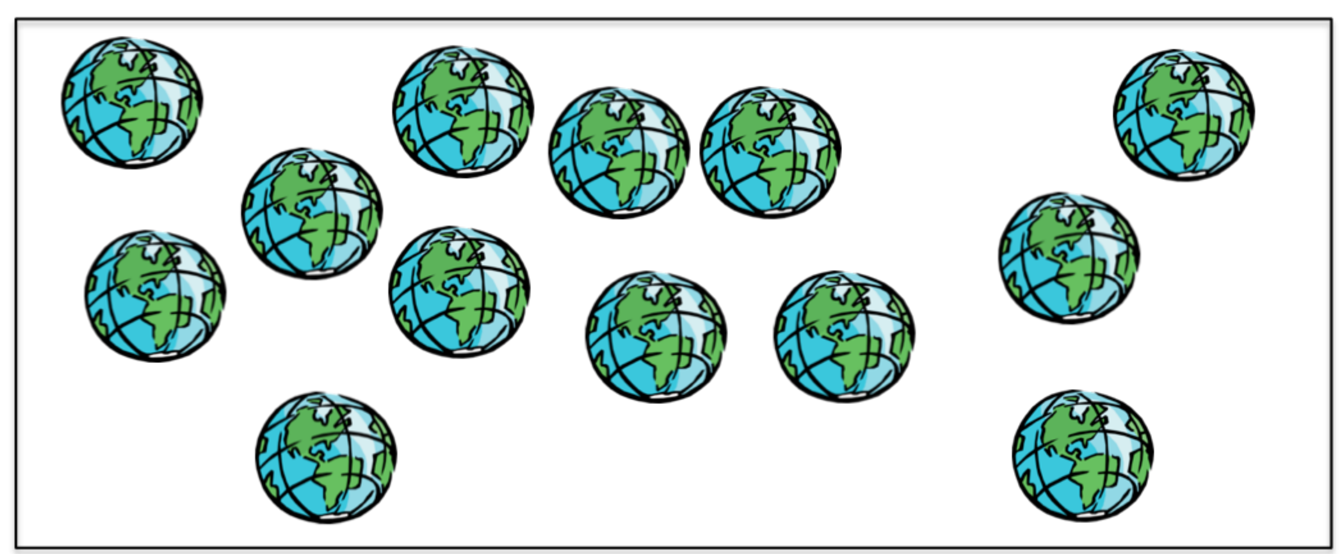
Generic code

Formal Language

- Syntax: What sentences are allowed?
- Semantics:
  - What are the possible worlds?
  - Which sentences are true in which worlds? (i.e., definition of truth)
- Model theory: how do we define whether a statement is true or not?
  - Truth and entailment
- Proof theory: what conclusion can we draw given a state of partial knowledge?
  - Soundness and completeness



Syntax



Semantics

Logic Language

- Natural language?
- Propositional logic
  - Syntax: $P \vee (\neg Q \wedge R)$; $X \Leftrightarrow (R \Rightarrow S)$
  - Possible model: {P=true, Q=true, R=false, S=true, X=true} or 11011
  - Possible world: interpretations of symbols
  - Semantics: $\alpha \wedge \beta$ is true in a world iff α is true and β is true (etc.)
- First-order logic
  - Syntax: $\forall x\ \exists y\ P(x,y) \wedge \neg Q(Joe,f(x)) \Rightarrow f(x)=f(y)$
  - Possible model: Objects o_1, o_2, o_3; P holds for <o_1,o_2>; Q holds for <o_3>; $f(o_1)=o_1$; $Joe=o_3$; etc.
  - Possible world: interpretations of objects, predicates, and functions.
  - Semantics: $\phi(\sigma)$ is true in a world if $\sigma=o_j$ and ϕ holds for o_j; etc.

Summary

- Single-agent
- World is deterministic
- State is partially-observable

- Planning agent instead of reflex agent
- Derives new facts from what it currently knows

Propositional Logic

Propositional Logic

- Symbol:
  - Variable that can be true or false
  - We'll try to use capital letters, e.g. A, B, $P_{1,2}$
  - Often include True and False
- Operators:
  - $\neg$ A: not A
  - A $\wedge$ B: A and B (conjunction)
  - A $\vee$ B: A or B (disjunction) Note: this is not an "exclusive or"
  - A $\Rightarrow$ B: A implies B (implication). If A then B
  - A $\Leftrightarrow$ B: A if and only if B (biconditional)
- Sentences

Propositional Logic Syntax

- Given: a set of proposition symbols $\{X_1, X_2, \ldots, X_n\}$
  - Sentence $\rightarrow$ AtomicSentence | ComplexSentence
  - AtomicSentence $\rightarrow$ True | False | Symbol
  - Symbol $\rightarrow X_1 \mid X_2 \mid \ldots \mid X_n$
  - ComplexSentence $\rightarrow$ $\neg$Sentence

    | (Sentence $\wedge$ Sentence)

    | (Sentence $\vee$ Sentence)

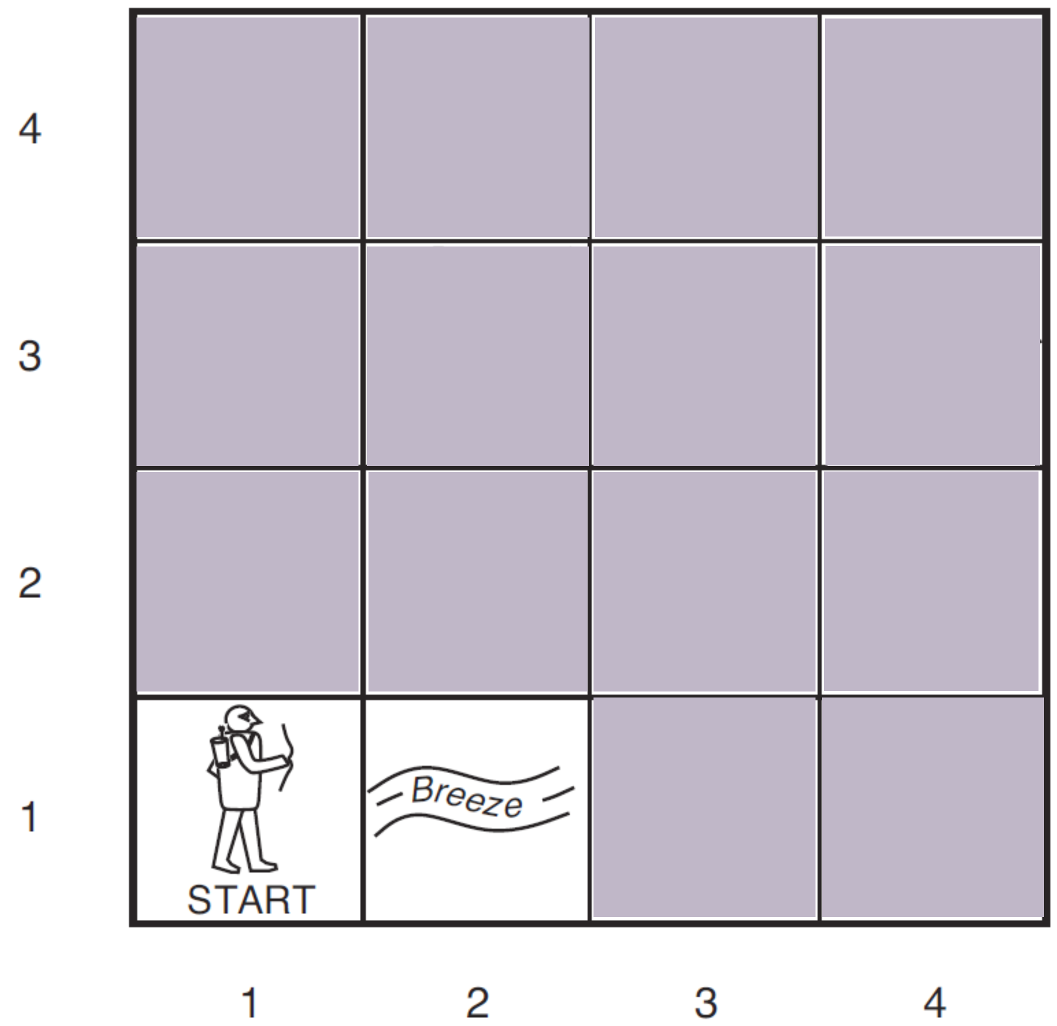
    | (Sentence $\Rightarrow$ Sentence)

    | (Sentence $\Leftrightarrow$ Sentence)

Example: Wumpus World

Logical Reasoning

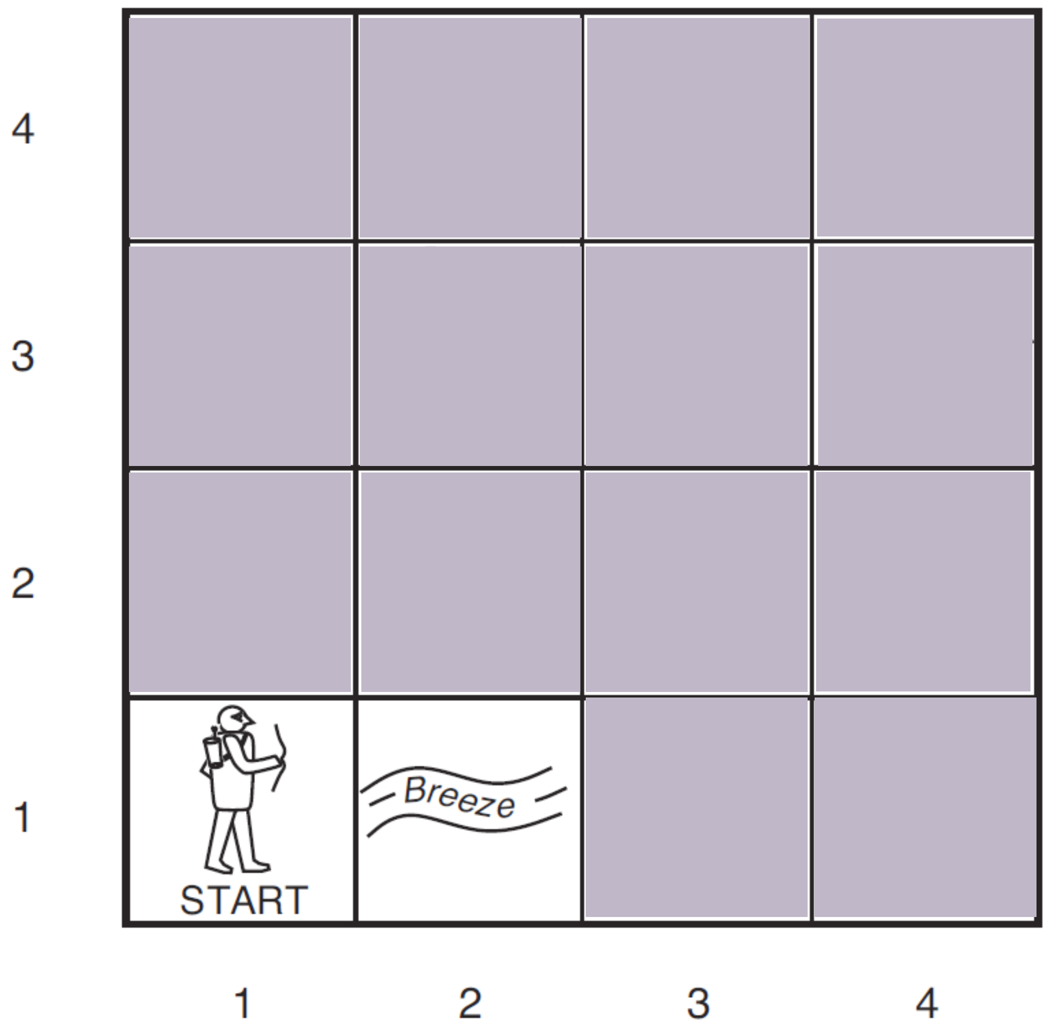
- B_{ij} = breeze felt
- S_{ij} = stench smelt
- P_{ij} = pit here
- W_{ij} = wumpus here
- G_{ij} = gold



http://thiagodnf.github.io/wumpus-world-simulator/

Wumpus World: Tell KB

- There is no pit in [1, 1]:
  - R1: $\neg P_{1,1}$
- A square is breezy iff there is a pit in a neighboring square:
  - R2: $B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$
  - R3: $B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$
  - …
- The first two percepts:
  - R4: $\neg B_{1,1}$
  - R5: $B_{2,1}$



Truth from Semantics

- A model specifies the truth value of every proposition symbol (e.g., P, $\neg P$, True, False)
- The truth value of complex sentences is defined in terms of the truth values of its elements:
  - $\neg P, P \wedge Q, P \vee Q, P \Rightarrow Q, P \Leftrightarrow Q$

Truth Tables

$\alpha \vee \beta$ is <u>inclusive or</u>, not exclusive

α	β	$\alpha \wedge \beta$
F	F	F
F	T	F
T	F	F
T	T	T

α	β	$\alpha \vee \beta$
F	F	F
F	T	T
T	F	T
T	T	T

Truth Tables

$\boldsymbol{\alpha \Rightarrow \beta}$ is equivalent to $\boldsymbol{\neg\alpha \vee \beta}$

α	β	$\alpha \Rightarrow \beta$	$\neg\alpha$	$\neg\alpha \vee \beta$
F	F	T	T	T
F	T	T	T	T
T	F	F	F	F
T	T	T	F	T

Truth Tables

$\boldsymbol{\alpha} \Leftrightarrow \boldsymbol{\beta}$ is equivalent to $(\boldsymbol{\alpha} \Rightarrow \boldsymbol{\beta}) \wedge (\boldsymbol{\beta} \Rightarrow \boldsymbol{\alpha})$

α	β	$\alpha \Leftrightarrow \beta$	$\alpha \Rightarrow \beta$	$\beta \Rightarrow \alpha$	$(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)$
F	F	T	T	T	T
F	T	F	T	F	F
T	F	F	F	T	F
T	T	T	T	T	T

Propositional Logic Semantics

function PL-TRUE?(α,model) **returns** true or false
  if α is a symbol **then return** Lookup(α, model)
  if Op(α) = $\neg$ **then return** not(PL-TRUE?(Arg1(α),model))
  if Op(α) = $\wedge$ **then return** and(PL-TRUE?(Arg1(α),model),
                 PL-TRUE?(Arg2(α),model))
  if Op(α) = $\Rightarrow$ **then return** or(PL-TRUE?(Arg1(α),model),
                 not(PL-TRUE?(Arg2(α),model)))
etc. (Sometimes called "recursion over syntax")

Logical Consequences

- Entailment: determines truth of sentence based on semantics (from outside)
- Inference: generates new sentence from current KB (from inside)
- Two closely related, but very different, concepts

Entailment

Entailment: $\alpha \vDash \beta$ ("α entails β" or "β follows from α") iff in every world where α is true, β is also true

- I.e., the α-worlds are a subset of the β-worlds [*models*(α) $\subseteq$ *models*(β)]

Usually we want to know if *KB* $\vDash$ *query*

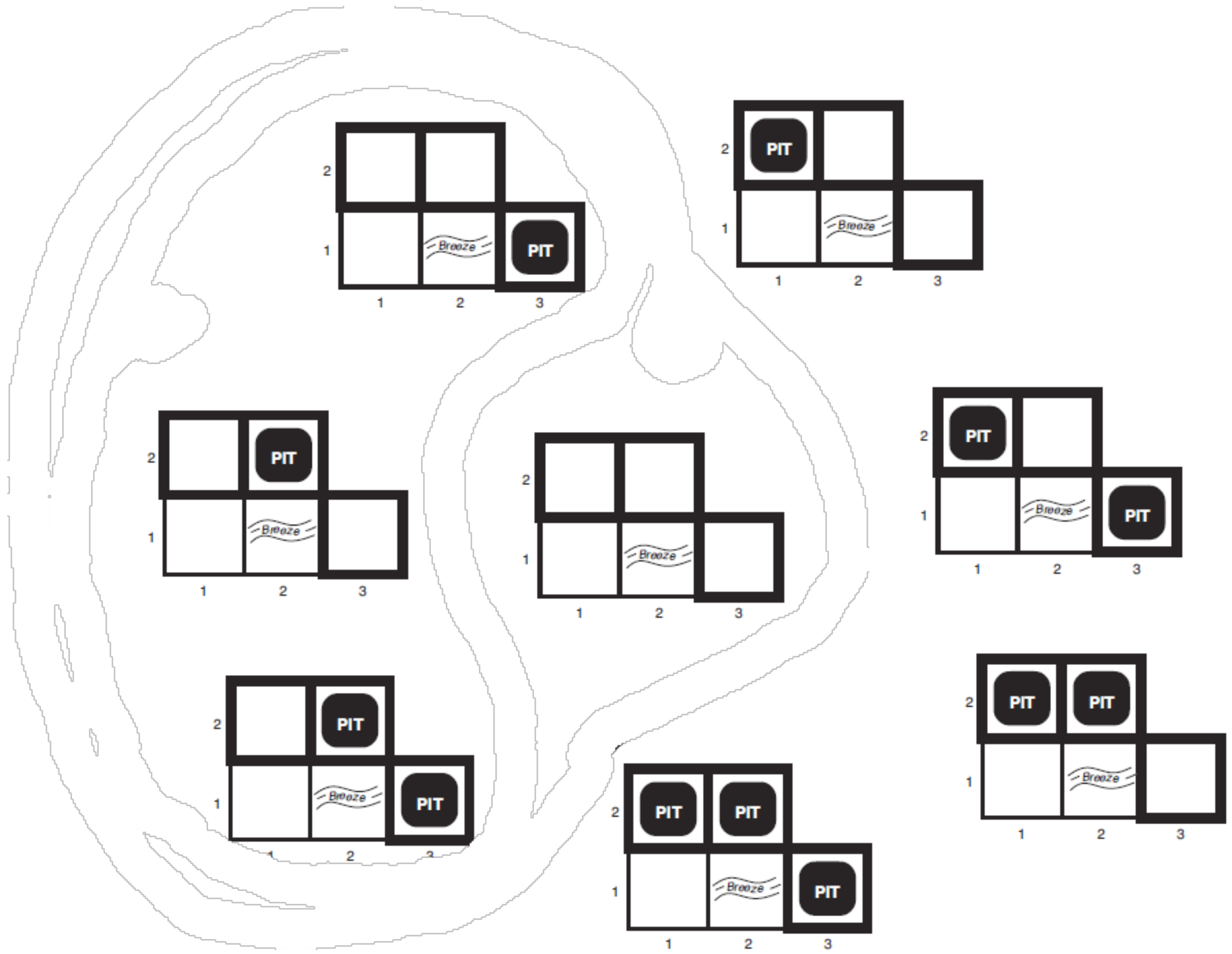
- *models*(*KB*) $\subseteq$ *models*(*query*)
- In other words
  - *KB* removes all impossible models (any model where *KB* is false)
  - If β is true in all of these remaining models, we conclude that β must be true

Entailment and implication are very much related

- However, entailment relates two sentences, while an implication is itself a sentence (usually derived via inference to show entailment)

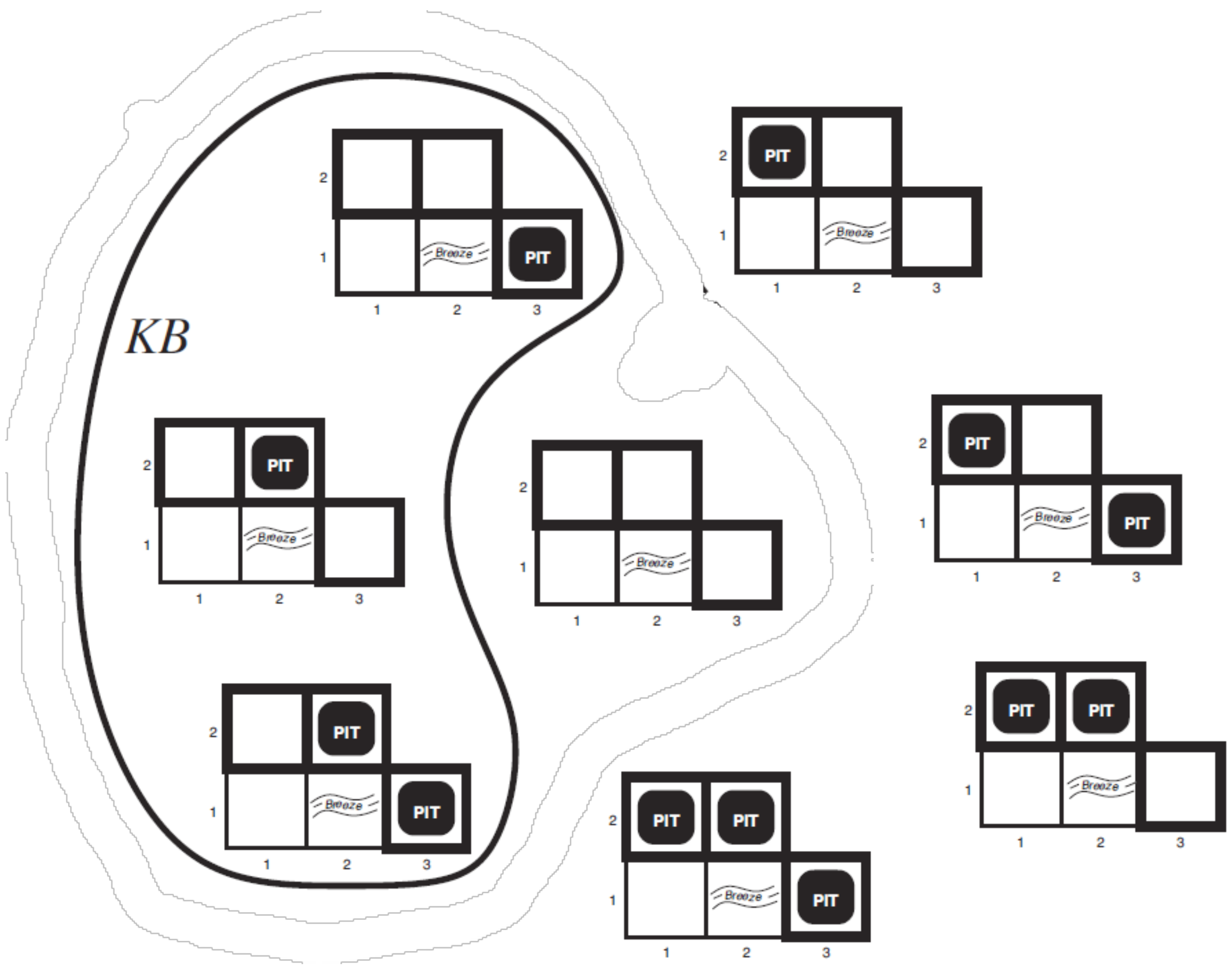
Wumpus World: Model

- Possible worlds/models
- $P_{1,2}$ $P_{2,2}$ $P_{3,1}$



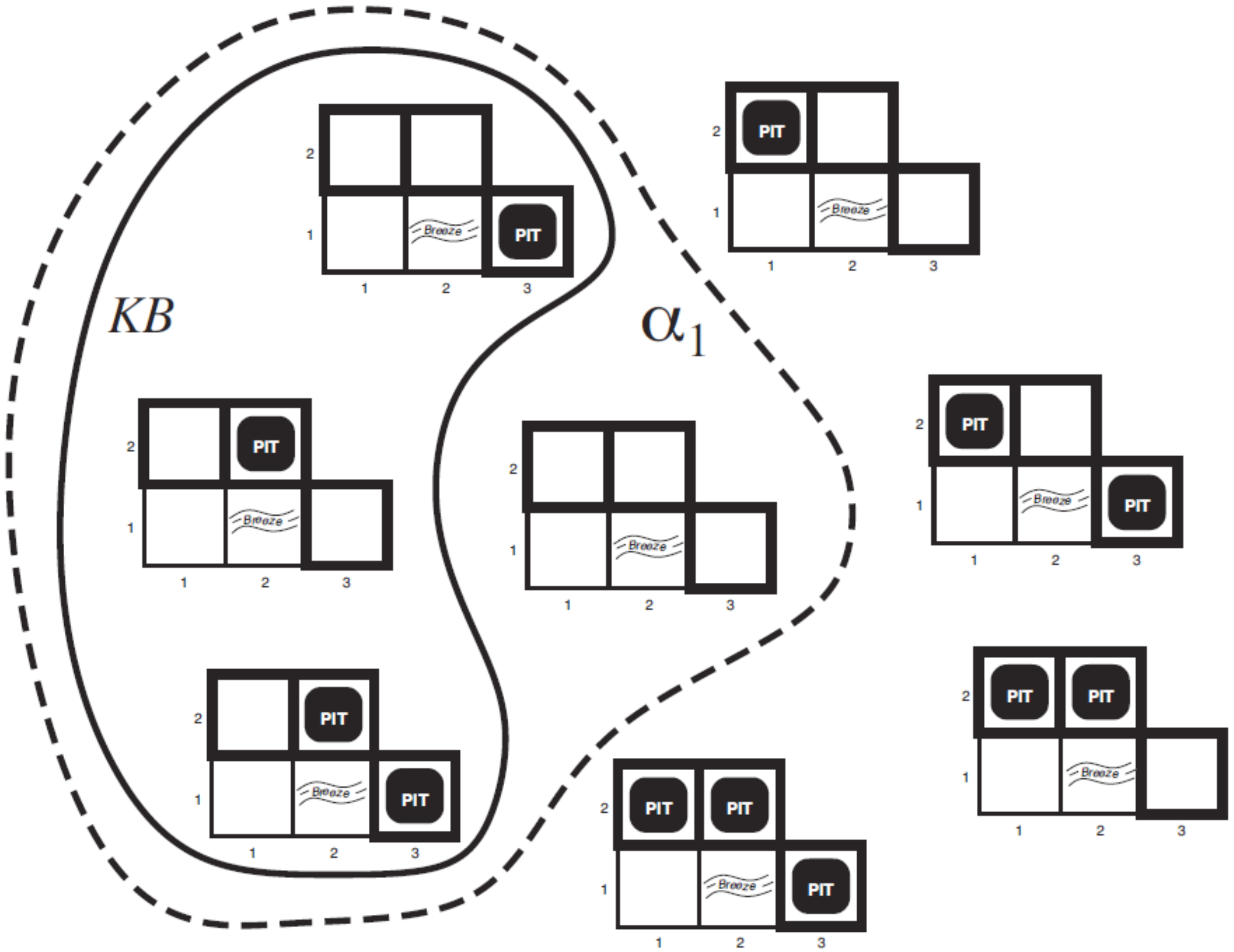
Wumpus World: KB

- Possible worlds/models
- $P_{1,2}$ $P_{2,2}$ $P_{3,1}$
- Knowledge base
  - Nothing in [1,1]
  - Breeze in [2,1]



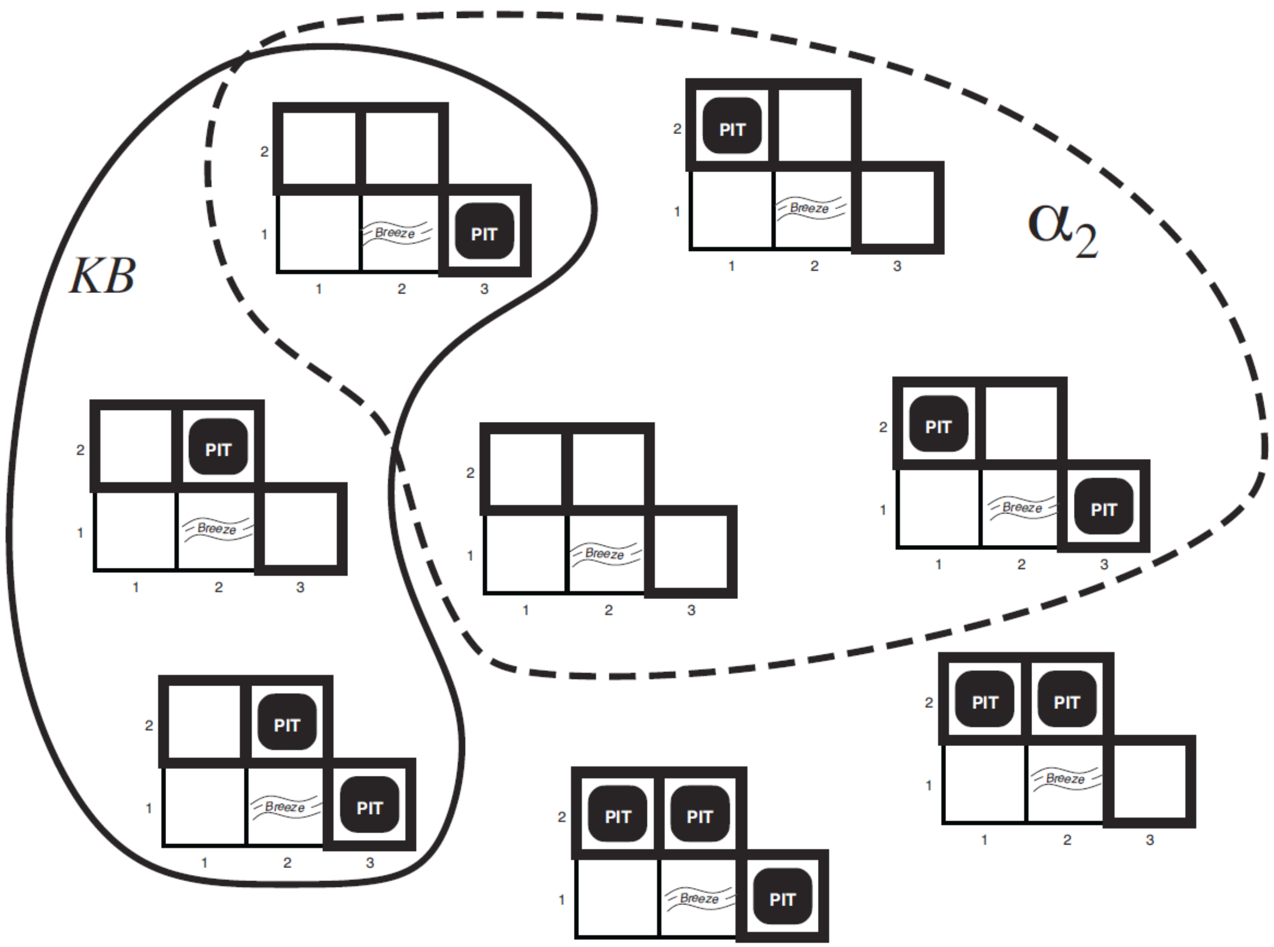
Wumpus World: Query 1

- Possible worlds/models
- $P_{1,2}$ $P_{2,2}$ $P_{3,1}$
- Knowledge base
  - Nothing in [1,1]
  - Breeze in [2,1]
- Query α_1:
  - No pit in [1,2]



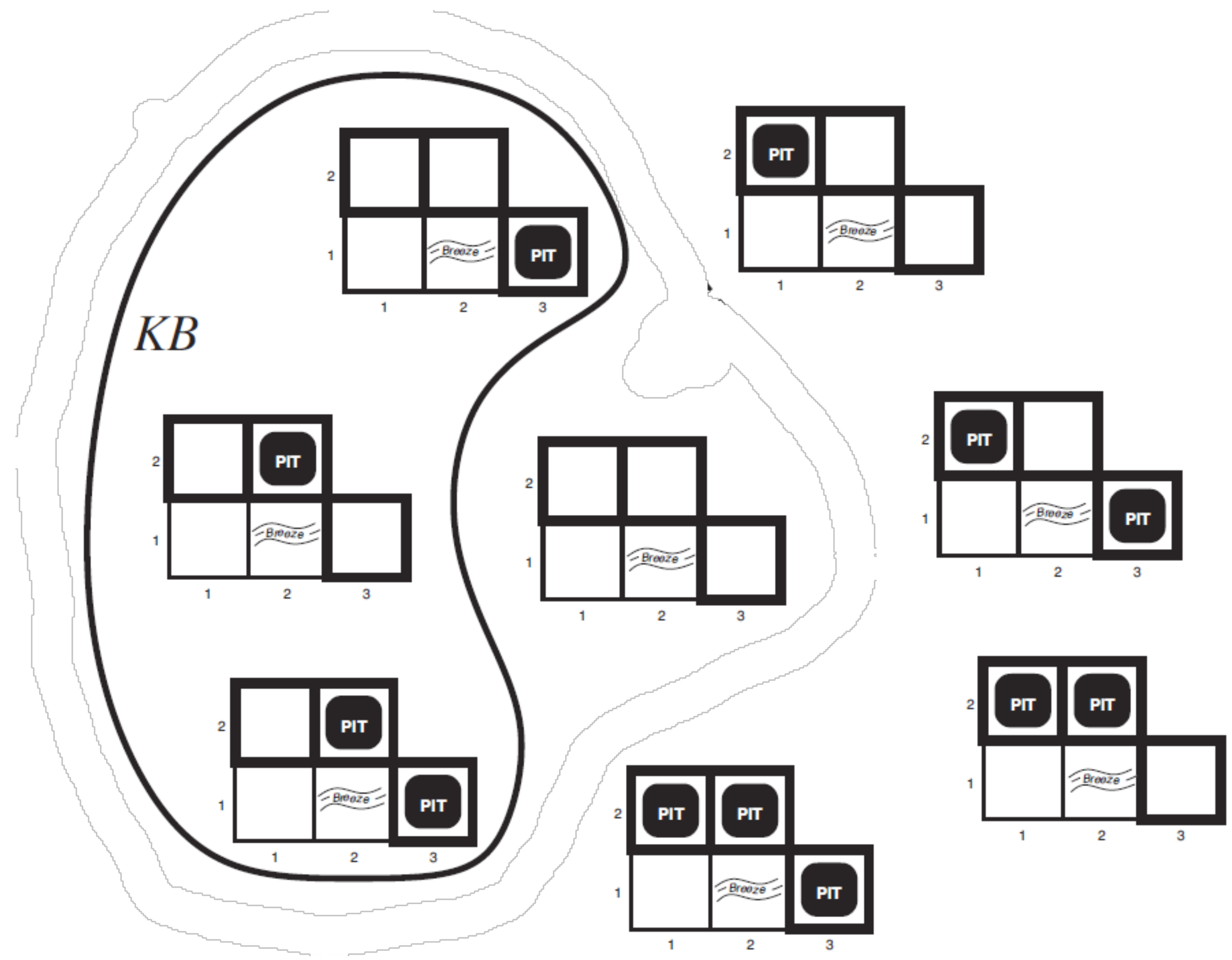
Wumpus World: Query 2

- Possible worlds/models
- $P_{1,2}$ $P_{2,2}$ $P_{3,1}$
- Knowledge base
  - Nothing in [1,1]
  - Breeze in [2,1]
- Query α_2:
  - No pit in [2,2]



Quiz: Wumpus World

- Possible worlds/models
- $P_{1,2}$ $P_{2,2}$ $P_{3,1}$
- Knowledge base
  - Nothing in [1,1]
  - Breeze in [2,1]
- Query α_3:
  - No pit in [3,1]



Sentences as Constraints

Adding a sentence to our knowledge base constrains the number of possible models:

KB: Nothing

Possible Models

P	Q	R
false	false	false
false	false	true
false	true	false
false	true	true
true	false	false
true	false	true
true	true	false
true	true	true

Sentences as Constraints

Adding a sentence to our knowledge base constrains the number of possible models:

KB: Nothing

KB: $[(P \land \neg Q) \lor (Q \land \neg P)] \Rightarrow R$

Possible Models	P	Q	R
	false	false	false
	false	false	true
	false	true	false
	false	true	true
	true	false	false
	true	false	true
	true	true	false
	true	true	true

Sentences as Constraints

Adding a sentence to our knowledge base constrains the number of possible models:

KB: Nothing

KB: $[(P \wedge \neg Q) \vee (Q \wedge \neg P)] \Rightarrow R$

KB: R, $[(P \wedge \neg Q) \vee (Q \wedge \neg P)] \Rightarrow R$

Possible Models

P	Q	R
false	false	false
false	false	true
false	true	false
false	true	true
true	false	false
true	false	true
true	true	false
true	true	true

Validity and Satisfiability

- A sentence is valid if it is true in every model
  - α entails β if and only if $\alpha \Rightarrow \beta$ is valid
  - A valid sentence is also called tautology
- A sentence is satisfiable if it is true in some model
- A sentence is unsatisfiable if it is true in no model

Logical Agents

Inference

Simple model checking
Efficient Model Checking via Satisfiability
Theorem proving

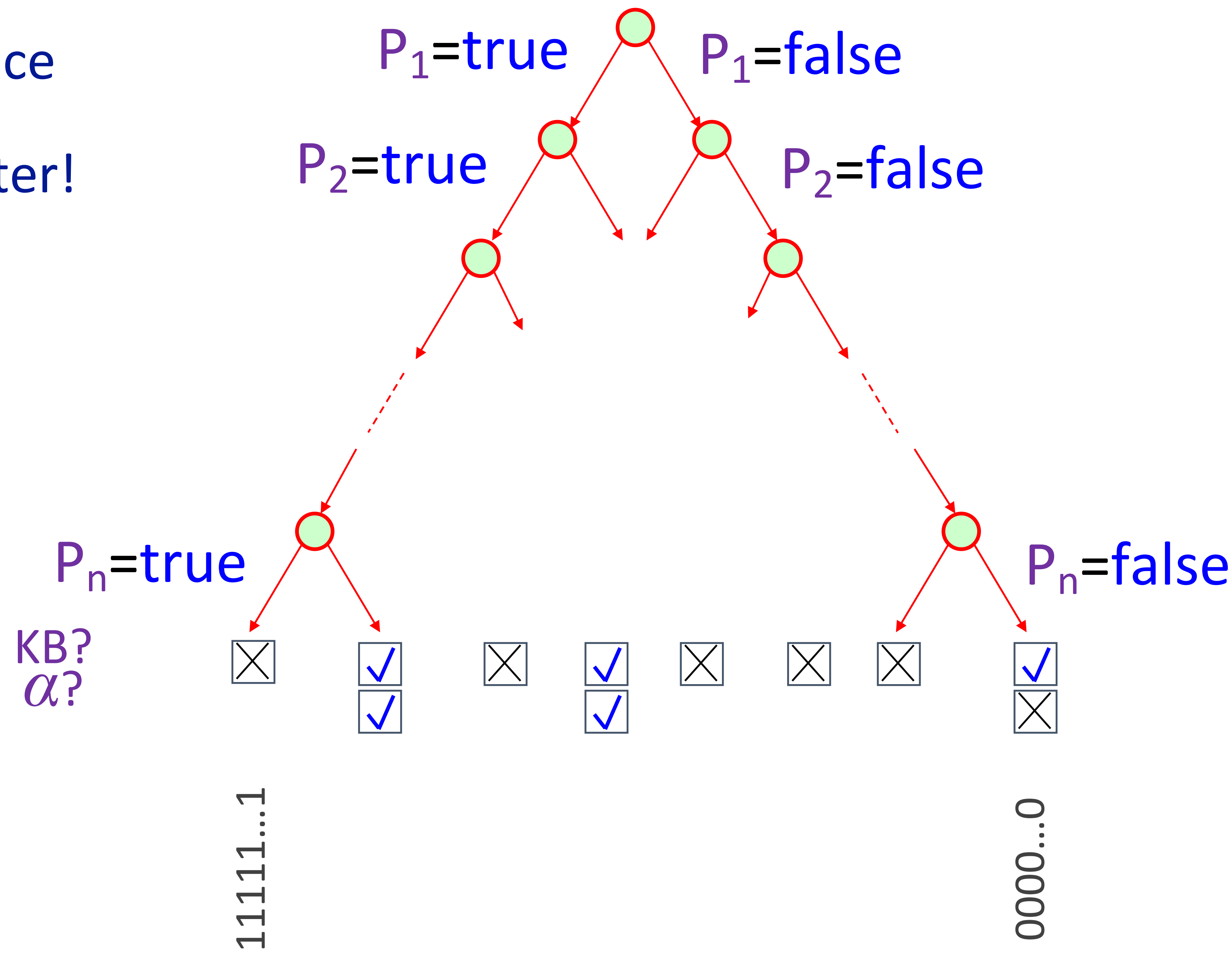
Simple Model Checking

```
function TT-ENTAILS?(KB, α) returns true or false
   return TT-CHECK-ALL(KB, α, symbols(KB) U symbols(α),{})

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
         P ← first(symbols)
         rest ← rest(symbols)
         return  and (TT-CHECK-ALL(KB, α, rest, model ∪ {P = true})
                          TT-CHECK-ALL(KB, α, rest, model ∪ {P = false }))
```

Simple Model Checking, contd.

- Same recursion as backtracking
- $O(2^n)$ time, linear space
- We can do much better!



Efficient Model Checking via Satisfiability

- Assume we have a hyper-efficient SAT solver; how can we use it to test entailment?
- Suppose $\alpha \vDash \beta$
- Then $\alpha \Rightarrow \beta$ is true in all worlds (Deduction theorem)
- Hence $\neg(\alpha \Rightarrow \beta)$ is false in all worlds
- Hence $\alpha \wedge \neg\beta$ is false in all worlds, i.e., unsatisfiable
- So, add the negated conclusion to what you know, test for (un)satisfiability; also known as reductio ad absurdum
- Efficient SAT solvers operate on conjunctive normal form

Conjunctive Normal Form (CNF)

- Every sentence can be expressed as a conjunction of clauses
- A clause is a disjunction of literals
- A literal is a symbol or a negated symbol
- Conversion to CNF by a sequence of standard transformations:
  - $B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$
  - $(B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})$
  - $(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge (\neg(P_{1,2} \vee P_{2,1}) \vee B_{1,1})$
  - $(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge ((\neg P_{1,2} \wedge \neg P_{2,1}) \vee B_{1,1})$
  - $(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge (\neg P_{1,2} \vee B_{1,1}) \wedge (\neg P_{2,1} \vee B_{1,1})$

Inference via Theorem Proving

- KB: set of sentences
- Inference rule specifies when:
  - If certain sentences belong to KB, you can add certain other sentences to KB
- Proof (KB $\vdash \alpha$) is a sequence of applications of inference rules starting from KB and ending in α
- Inference is a completely mechanical operation guided by syntax, no reference to possible worlds

Example of Inference Rules

- Modus ponens: $\frac{\alpha \Rightarrow \beta, \alpha}{\beta}$
- And elimination: $\frac{\alpha \wedge \beta}{\alpha}$
- Biconditional elimination: $\frac{\alpha \Leftrightarrow \beta}{(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)}$

Soundness and Completeness

- We want inference to be sound:
  - If we can prove B from A ($A \vdash B$), then $A \vDash B$
- We would like inference to be complete:
  - If $A \vDash B$, then we can prove B from A ($A \vdash B$)
- These are properties of the relationship between proof and truth.

PL is Sound and Complete!

- Theorem: Sound and complete inference can be achieved in PL with one rule: resolution
  - $\frac{\alpha \vee \beta, \neg \beta}{\alpha}$
  - More generally, $\frac{\alpha \vee \beta, \neg \beta \vee \gamma}{\alpha \vee \gamma}$
  - More generally yet, $\frac{\alpha_1 \vee \cdots \vee \alpha_n \vee \beta, \neg \beta \vee \gamma_1 \vee \cdots \gamma_m}{\alpha_1 \vee \cdots \vee \alpha_n \vee \gamma_1 \vee \cdots \gamma_m}$
- KB assumed to be in CNF
- Show KB $\vDash \alpha$ by showing unsatisfiability of (KB $\wedge \neg \alpha$)