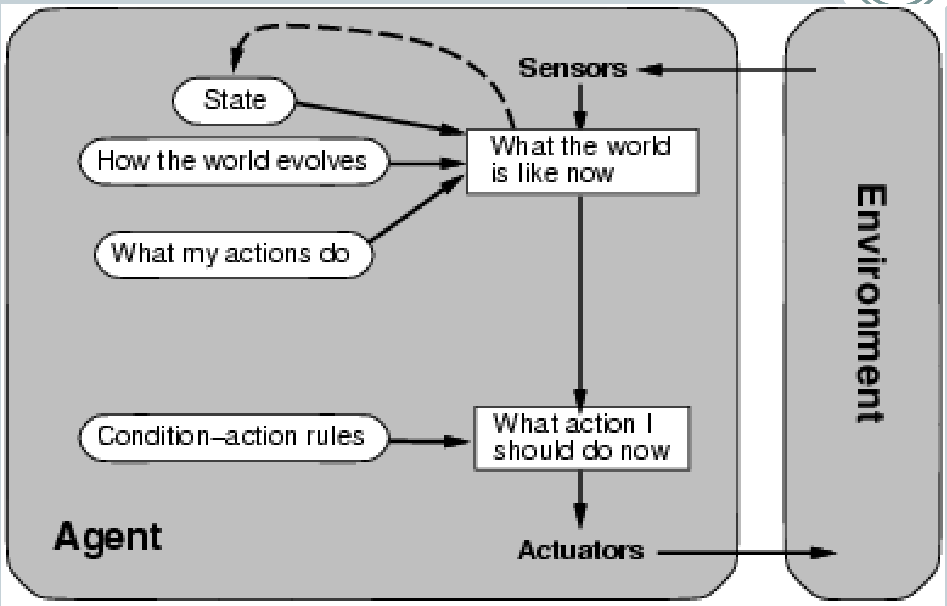
Logic Agents and Propositional Logic

Model-based Agents

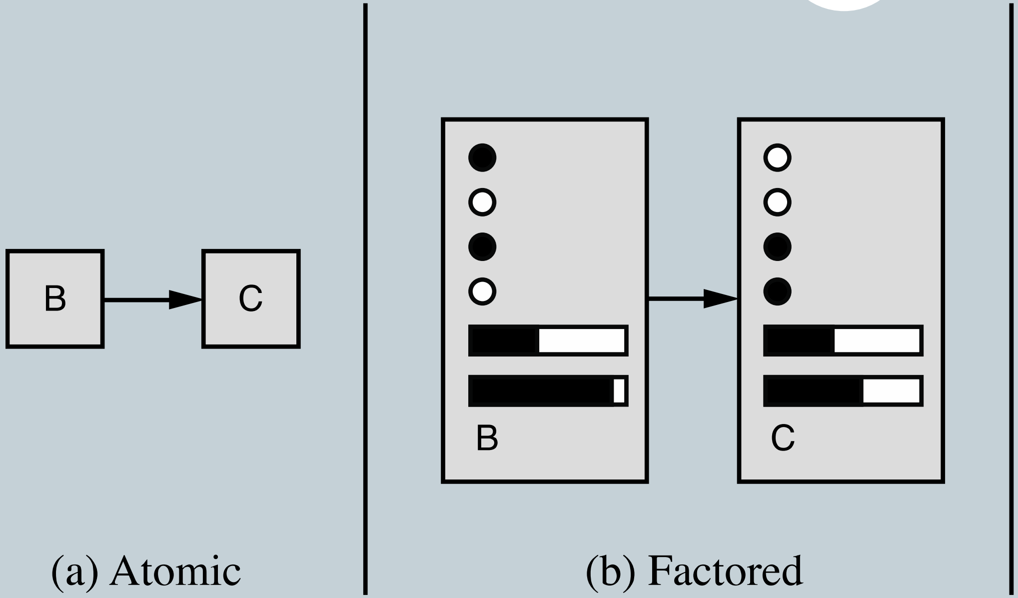


- Know how world evolves
  - Overtaking car gets closer from behind
- How agents actions affect the world
  - Wheel turned clockwise takes you right
- Model base agents update their state.
- Can also add goals and utility/performance measures.

Knowledge Representation Issues

- The Relevance Problem.
- The completeness problem.
- The Inference Problem.
- The Decision Problem.
- The Robustness problem.

Agent Architecture: Logical Agents



A model is a **structured** representation of the world.

• Graph-Based Search: State is **black box**, no internal structure, atomic.
• Factored Representation: State is list or vector of facts.
• Facts are expressed in **formal logic.**

Limitations of CSPs

- Constraint Satisfaction Graphs can represent much information about an agent's domain.
- Inference can be a powerful addition to search (arc consistency).
- Limitations of expressiveness:
  - Difficult to specify complex constraints, arity > 2.
  - Make explicit the form of constraints (<>, implies…).
- Limitations of Inference with Arc consistency:
  - Non-binary constraints.
  - Inferences involving multiple variables.

Logic: Motivation

- 1^{st}-order logic is highly expressive.
  - Almost all of known mathematics.
  - All information in relational databases.
  - Can translate much natural language.
  - Can reason about other agents, beliefs, intentions, desires…
- Logic has **complete** inference procedures.
  - All valid inferences can be proven, in principle, by a machine.
- Cook’s fundamental theorem of NP-completeness states that all difficult search problems (scheduling, planning, CSP etc.) can be represented as logical inference problems. (U of T).

Logic vs. Programming Languages

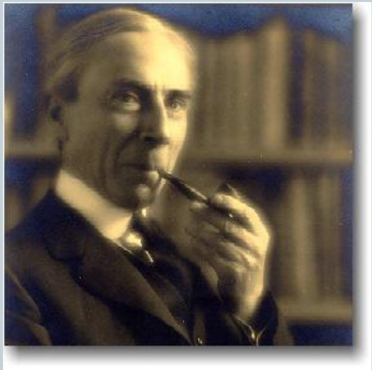
- Logic is declarative.
- Think of logic as a kind of **language** for expressing knowledge.
  - Precise, computer readable.
- A proof system allows a computer to **infer** consequences of known facts.
- Programming languages lack general mechanism for deriving facts from other facts. Traffic Rule Demo

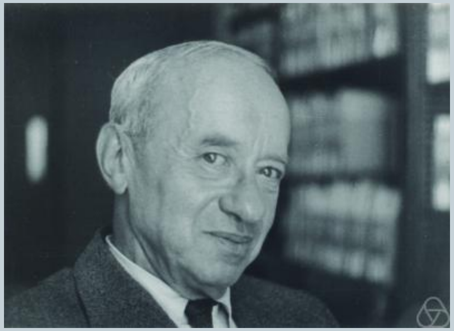
Logic and Ontologies

- Large collections of facts in logic are structured in hierarchices known as **ontologies**
  - See chapter in textbook, we' re skipping it.
- Cyc: Large Ontology Example.
- Cyc Ontology Hierarchy.
- Cyc Concepts Lookup
  - E.g., games, Vancouver.

1st-order Logic: Key ideas

- The fundamental question: *What kinds of information do we need to represent?* (Russell, Tarski).
- The world/environment consists of
  - Individuals/entities.
  - Relationships/links among them.



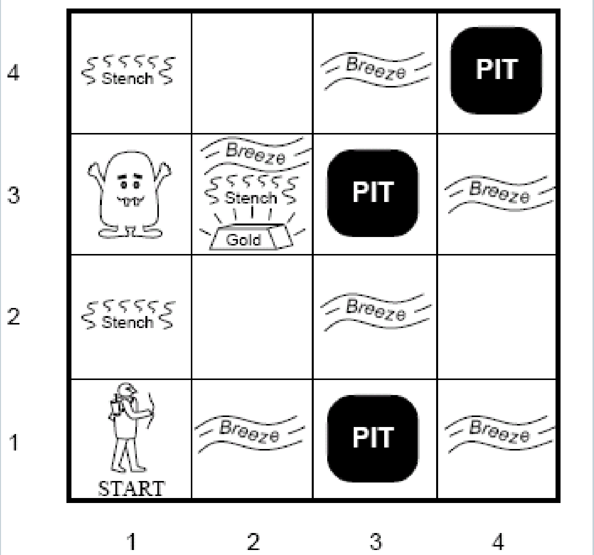


Knowledge-Based Agents

- KB = knowledge base
  - A set of sentences or facts
  - e.g., a set of statements in a logic language
- Inference
  - Deriving new sentences from old
  - e.g., using a set of logical statements to infer new ones
- A simple model for reasoning
  - Agent is told or perceives new evidence
    - E.g., A is true
  - Agent then infers new facts to add to the KB
    - E.g., KB = { A -> (B OR C) }, then given A and not C we can infer that B is true
    - B is now added to the KB even though it was not explicitly asserted, i.e., the agent inferred B

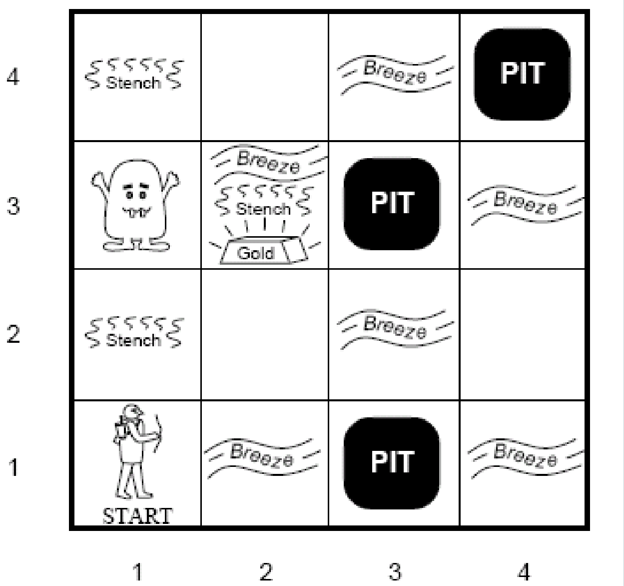
Wumpus World

- Environment
  - Cave of 4 × 4
  - Agent enters in [1,1]
  - 16 rooms
    - Wumpus: A deadly beast who kills anyone entering his room.
    - Pits: Bottomless pits that will trap you forever.
    - Gold



Wumpus World

- Agents Sensors:
  - Stench next to Wumpus
  - Breeze next to pit
  - Glitter in square with gold
  - Bump when agent moves into a wall
  - Scream from wumpus when killed
- Agents actions
  - Agent can move forward, turn left or turn right
  - Shoot, one shot



What is a logical language?

- A formal language
  - KB = set of sentences

- Syntax
  - what sentences are legal (well-formed)
  - E.g., arithmetic
    - X+2 >= y is a wf sentence, +x2y is not a wf sentence

- Semantics
  - loose meaning: the interpretation of each sentence
  - More precisely:
    - Defines the truth of each sentence wrt to each possible world
  - e.g,
    - X+2 = y is true in a world where x=7 and y =9
    - X+2 = y is false in a world where x=7 and y =1

  - Note: standard logic – each sentence is T of F wrt eachworld
    - Fuzzy logic – allows for degrees of truth.

Propositional logic: Syntax

- Propositional logic is the simplest logic – illustrates basic ideas

- Atomic sentences = single proposition symbols
  - E.g., P, Q, R
  - Special cases: True = always true, False = always false

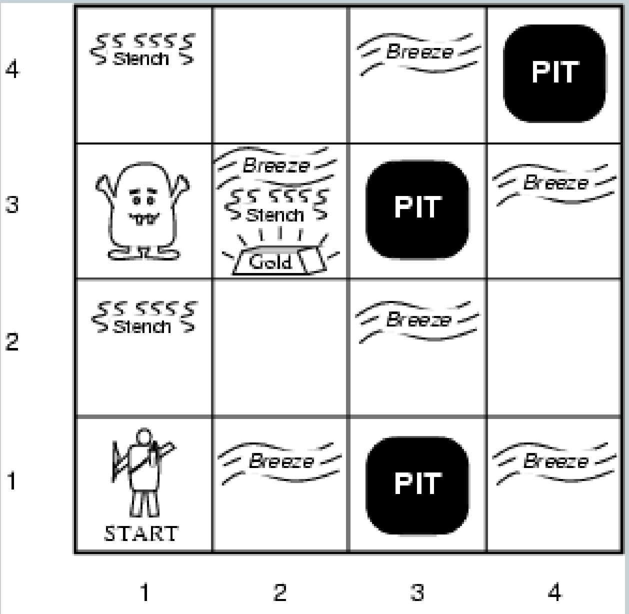
- Complex sentences:

  - 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)

Wumpus world sentences

Let $P_{i,j}$ be true if there is a pit in [i, j].
Let $B_{i,j}$ be true if there is a breeze in [i, j].

start: $\neg P_{1,1}$
$\neg B_{1,1}$
$B_{2,1}$



- "Pits cause breezes in adjacent squares"

  $B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$
  $B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$

- KB can be expressed as the conjunction of all of these sentences

- Note that these sentences are rather long-winded!
  - E.g., breeze "rule" must be stated explicitly for each square
  - First-order logic will allow us to define more general patterns.

Propositional logic: Semantics

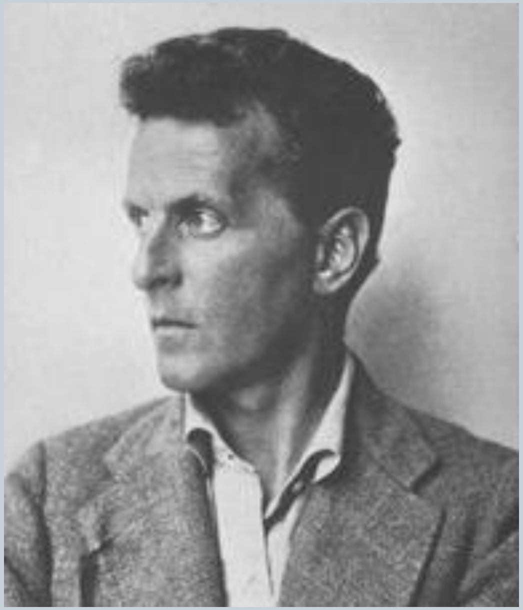
- A sentence is interpreted in terms of **models**, or **possible worlds**.
- These are formal structures that specify a truth value for **each sentence** in a consistent manner.

Ludwig Wittgenstein (1918):

1. The world is everything that is the case.

1.1 The world is the complete collection of facts, not of things.

1.11 The world is determined by the facts, and by being the *complete* collection of facts.



More on Possible Worlds

- m is a model of a sentence α if α is true in m
- $M(\alpha)$ is the set of all models of α
- Possible worlds ~ models
  - Possible worlds: potentially real environments
  - Models: mathematical abstractions that establish the truth or falsity of every sentence
- Example:
  - x + y = 4, where x = #men, y = #women
  - Possible models = all possible assignments of integers to x and y.

  - For CSPs, possible model = complete assignment of values to variables.
  - Wumpus Example Assignment style

Propositional logic: Formal Semantics

Each model/world specifies true or 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 **every** sentence, e.g.,

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

Truth tables for connectives

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

Truth tables for connectives

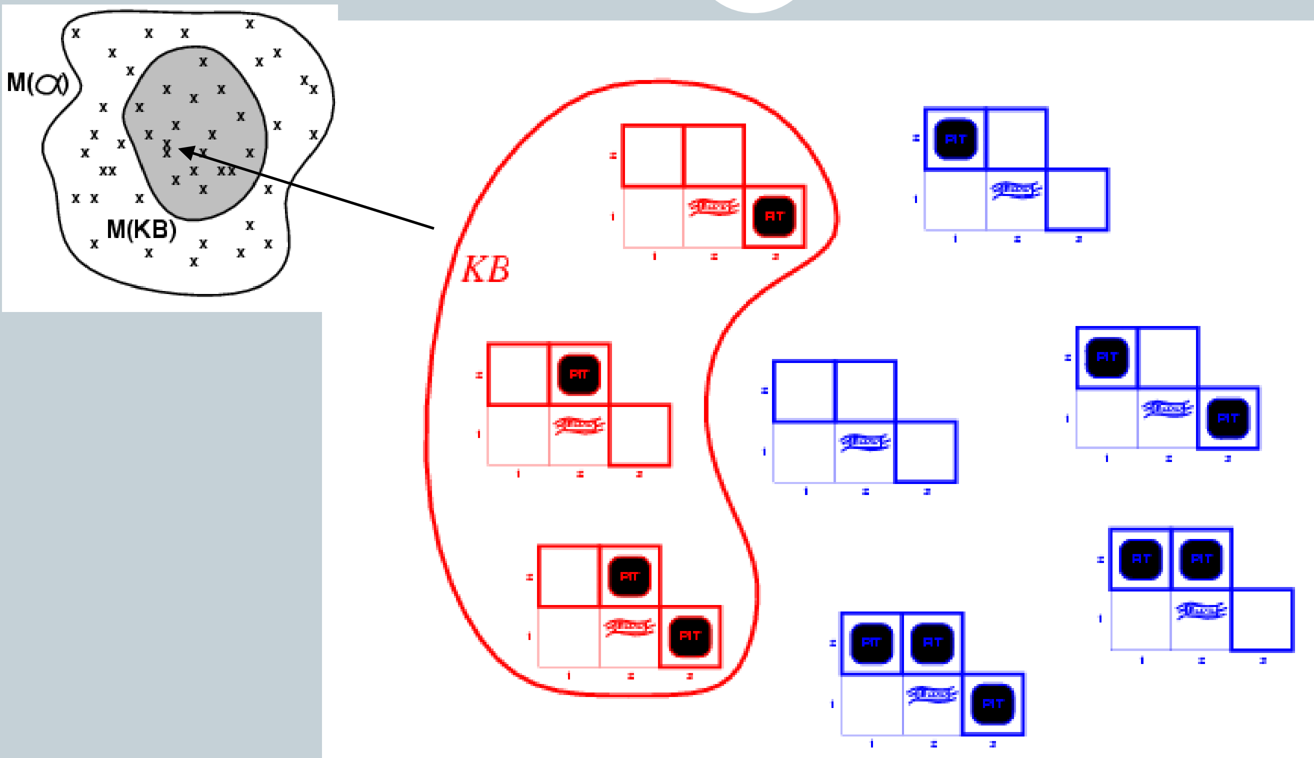
Evaluation Demo - Tarki's World

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

Implication is always true when the premise is false

Why? P=>Q means "if P is true then I am claiming that Q is true otherwise no claim"
Only way for this to be false is if P is true and Q is false

Wumpus models



- *KB* = all possible wumpus-worlds consistent with the observations and the "physics" of the Wumpus world.

Listing of possible worlds for the Wumpus KB

α_1 = ”square [1,2] is safe”.
KB = detect nothing in [1,1], detect breeze in [2,1]

$B_{1,1}$	$B_{2,1}$	$P_{1,1}$	$P_{1,2}$	$P_{2,1}$	$P_{2,2}$	$P_{3,1}$	KB	α_1
false	*false*	*false*	*false*	*false*	*false*	*false*	*false*	*true*
false	*false*	*false*	*false*	*false*	*false*	*true*	*false*	*true*
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
false	*true*	*false*	*false*	*false*	*false*	*false*	*false*	*true*
false	*true*	*false*	*false*	*false*	*false*	*true*	*true*	*true*
false	*true*	*false*	*false*	*false*	*true*	*false*	*true*	*true*
false	*true*	*false*	*false*	*false*	*true*	*true*	*true*	*true*
false	*true*	*false*	*false*	*true*	*false*	*false*	*false*	*true*
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
true	*true*	*true*	*true*	*true*	*true*	*true*	*false*	*false*

Entailment

- One sentence follows logically from another

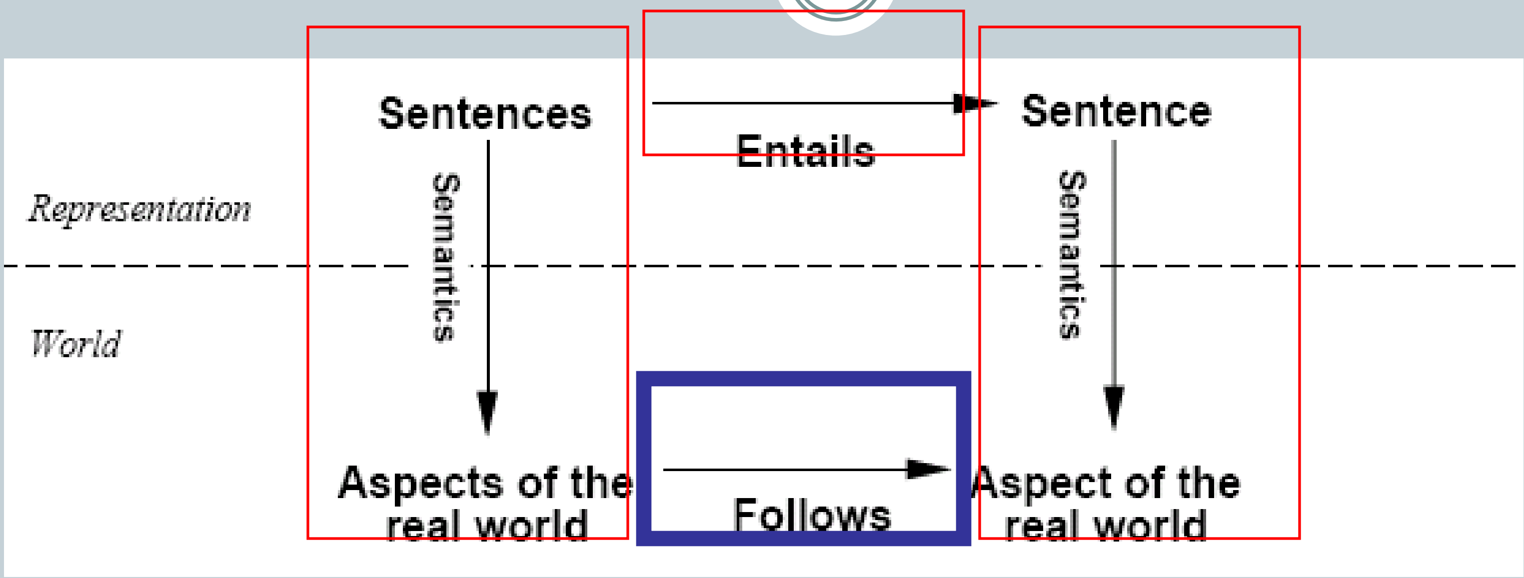
  $$\alpha \models \beta$$

  α entails sentence β *if and only if* β is true in all worlds where α is true.

  e.g., x+y=4 |= 4=x+y

- Entailment is a relationship between sentences that is based on semantics.

Schematic perspective

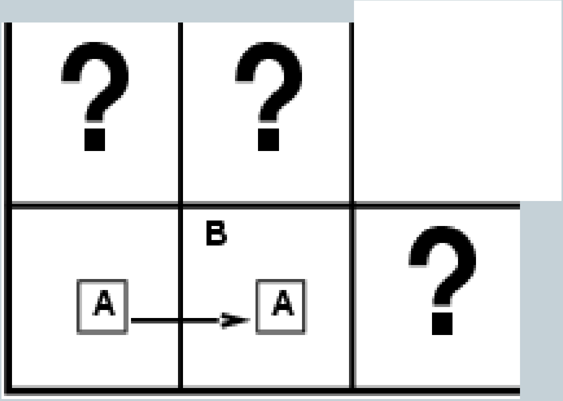


If KB is true in the real world, then any sentence α derived from KB by a sound inference procedure is also true in the real world.

Entailment in the wumpus world

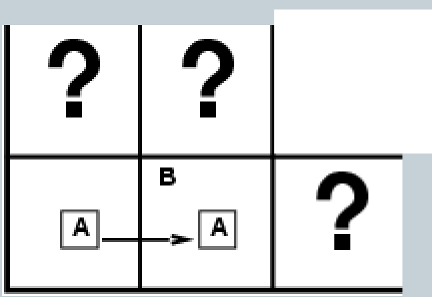
- Consider possible models for *KB* assuming only pits and a reduced Wumpus world

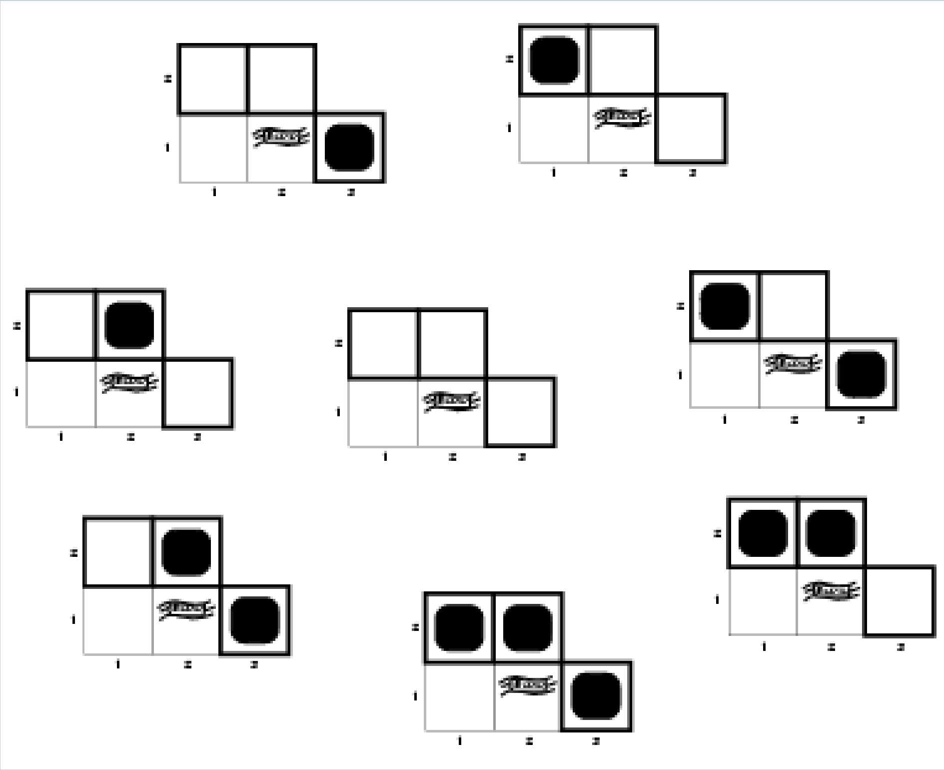
- Situation after detecting nothing in [1,1], moving right, detecting breeze in [2,1]



Wumpus models

All possible models in this reduced Wumpus world.

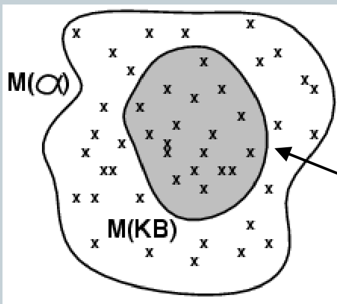


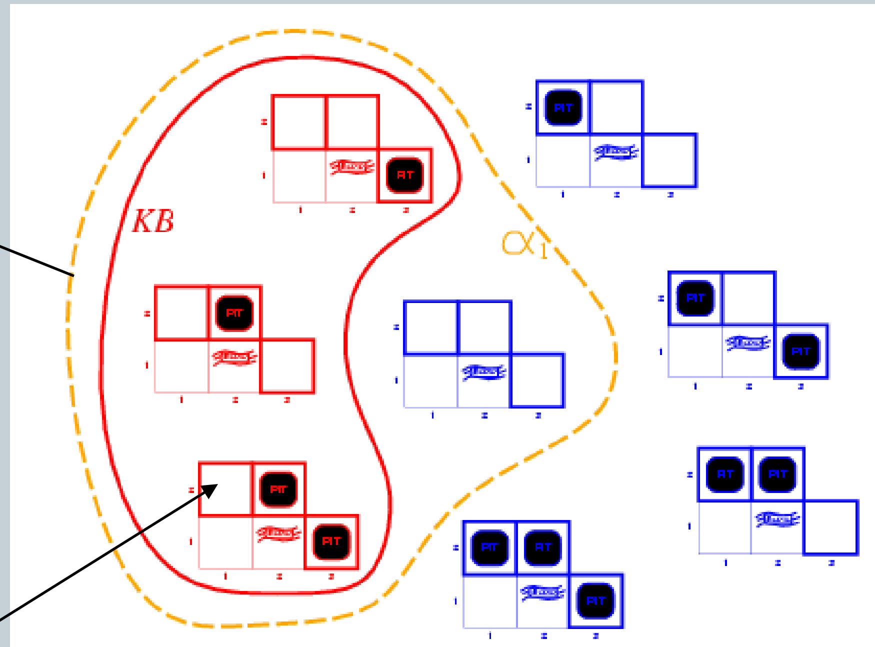


Inferring conclusions

- Consider 2 possible conclusions given a KB
  - α_1 = "[1,2] is safe"
  - α_2 = "[2,2] is safe“
- One possible inference procedure
  - Start with KB
  - Model-checking
    - Check if KB $\models \alpha$ by checking if in all possible models where KB is true that α is also true
- Comments:
  - Model-checking enumerates all possible worlds
    - Only works on finite domains, will suffer from exponential growth of possible models

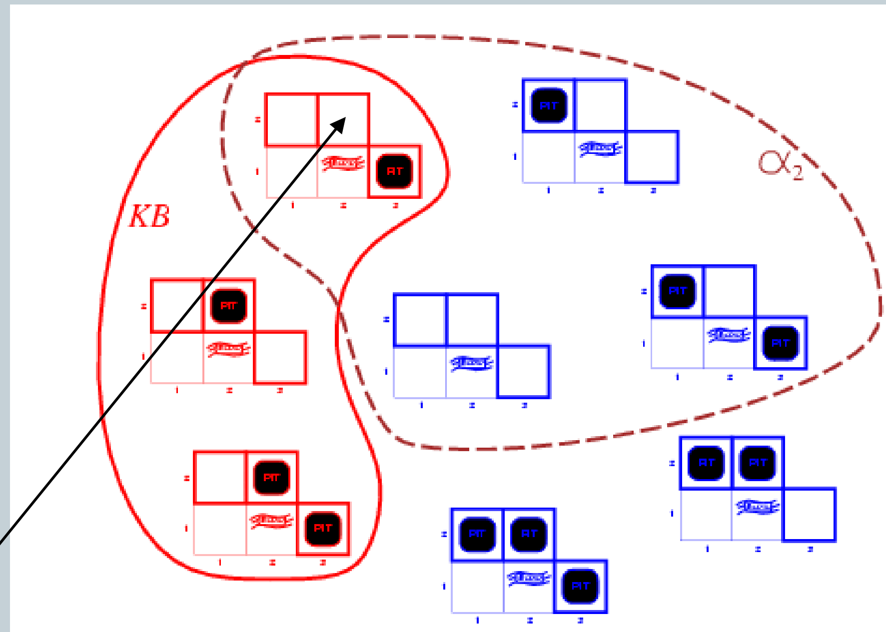
Wumpus models





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

Wumpus models



α_2 = "[2,2] is safe", $KB \not\models \alpha_2$

- There are some models entailed by KB where α_2 is false.
  - Wumpus Example Assignment style

Logical inference

- The notion of entailment can be used for inference.
  - Model checking (see wumpus example): enumerate all possible models and check whether α is true.

- If an algorithm only derives entailed sentences it is called *sound* or *truth preserving*.
- A proof system is **sound** if whenever the system derives α from KB, it is also true that $KB \models \alpha$
  - *E.g., model-checking is sound*

- Completeness : the algorithm can derive any sentence that is entailed.
- A proof system is **complete** if whenever $KB \models \alpha$, the system derives α from KB.

Inference by enumeration

- We want to see if α is entailed by KB
- Enumeration of all models is sound and complete.
- But…for *n* symbols, time complexity is $O(2^n)$…
- We need a more efficient way to do inference
  - But worst-case complexity will remain exponential for propositional logic

Logical equivalence

- To manipulate logical sentences we need some rewrite rules.
- Two sentences are logically equivalent iff they are true in same models: $\alpha \equiv \beta$ iff $\alpha \models \beta$ and $\beta \models \alpha$

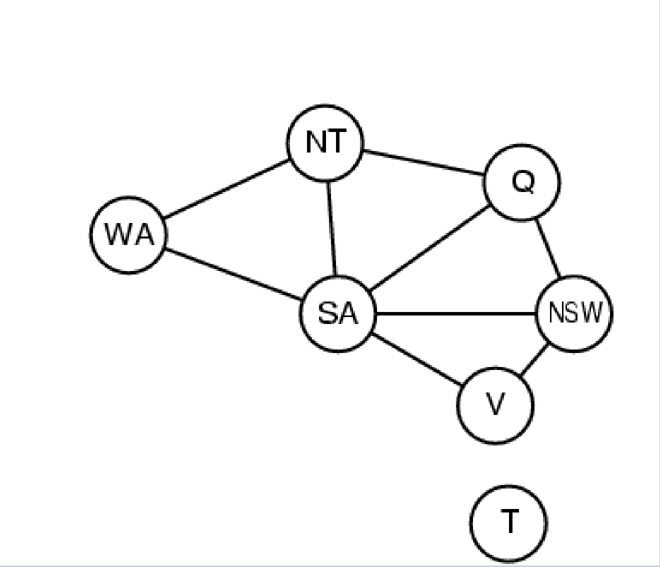
$(\alpha \land \beta)$	$\equiv$	$(\beta \land \alpha)$	commutativity of $\land$
$(\alpha \lor \beta)$	$\equiv$	$(\beta \lor \alpha)$	commutativity of $\lor$
$((\alpha \land \beta) \land \gamma)$	$\equiv$	$(\alpha \land (\beta \land \gamma))$	associativity of $\land$
$((\alpha \lor \beta) \lor \gamma)$	$\equiv$	$(\alpha \lor (\beta \lor \gamma))$	associativity of $\lor$
$\neg(\neg\alpha)$	$\equiv$	α	double-negation elimination
$(\alpha \Rightarrow \beta)$	$\equiv$	$(\neg\beta \Rightarrow \neg\alpha)$	contraposition
$(\alpha \Rightarrow \beta)$	$\equiv$	$(\neg\alpha \lor \beta)$	implication elimination
$(\alpha \Leftrightarrow \beta)$	$\equiv$	$((\alpha \Rightarrow \beta) \land (\beta \Rightarrow \alpha))$	biconditional elimination
$\neg(\alpha \land \beta)$	$\equiv$	$(\neg\alpha \lor \neg\beta)$	de Morgan
$\neg(\alpha \lor \beta)$	$\equiv$	$(\neg\alpha \land \neg\beta)$	de Morgan
$(\alpha \land (\beta \lor \gamma))$	$\equiv$	$((\alpha \land \beta) \lor (\alpha \land \gamma))$	distributivity of $\land$ over $\lor$
$(\alpha \lor (\beta \land \gamma))$	$\equiv$	$((\alpha \lor \beta) \land (\alpha \lor \gamma))$	distributivity of $\lor$ over $\land$

Exercises

- Show that *P implies Q*
  is logically equivalent to *(not P) or Q*.
  That is, one of these formulas is true in a model just in case the other is true.
- A **literal** is a formula of the form P or of the form not P, where P is an atomic formula. Show that the formula *(P or Q) and (not R)* has an equivalent formula that is a disjunction of a conjunction of literals. Thus the equivalent formula looks like this:
  [literal 1 and literal 2 and ….] or [literal 3 and …]

Propositional Logic vs. CSPs

- CSPs (Constraint Satisfaction Problem) are a special case as follows.
- The atomic formulas are of the type Variable = value.
- E.g., (WA = green).
- Negative constraints correspond to negated conjunctions.
- E.g. not (WA = green and NT = green).



Exercise: Show that every (binary) CSP is equivalent to a conjunction of literal disjunctions of the form [variable 1 = value 1 or variable 1 = value 2 or variable 2 = value 2 or ….] and […]

Normal Clausal Form

Eventually we want to prove: Knowledge base KB entails sentence α

We first rewrite into conjunctive normal form (CNF).

A "conjunction of disjunctions"

literals

$$\underbrace{(A \vee \neg B)}_{\text{Clause}} \wedge \underbrace{(B \vee \neg C \vee \neg D)}_{\text{Clause}}$$

- **Theorem: Any KB can be converted into an equivalent CNF**.
- k-CNF: exactly k literals per clause

Example: Conversion to CNF

$B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$

1. Eliminate $\Leftrightarrow$, replacing $\alpha \Leftrightarrow \beta$ with $(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)$.
   $(B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})$

2. Eliminate $\Rightarrow$, replacing $\alpha \Rightarrow \beta$ with $\neg\alpha \vee \beta$.
   $(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge (\neg(P_{1,2} \vee P_{2,1}) \vee B_{1,1})$

3. Move $\neg$ inwards using de Morgan's rules and double-negation:
   $(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge ((\neg P_{1,2} \wedge \neg P_{2,1}) \vee B_{1,1})$

4. Apply distributive law ($\wedge$ over $\vee$) and flatten:
   $(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge (\neg P_{1,2} \vee B_{1,1}) \wedge (\neg P_{2,1} \vee B_{1,1})$

Horn Clauses

Horn Clause = A clause with at most 1 positive literal.

e.g. $A \vee \neg B \vee \neg C$

- Every Horn clause can be rewritten as an implication with a conjunction of positive literals in the premises and at most a single positive literal as a conclusion.

e.g. $B \wedge C \Rightarrow A$

- 1 positive literal: definite clause

- 0 positive literals: Fact or integrity constraint:

  e.g. $(\neg A \vee \neg B) \equiv (A \wedge B \Rightarrow False)$

- Psychologically natural: a condition implies (causes) a single fact.

- The basis of **logic programming** (the prolog language).

SWI Prolog. Prolog and the Semantic Web. Prolog Applications

Summary

- Logical agents apply inference to a knowledge base to derive new information and make decisions
- Basic concepts of logic:
  - syntax: formal structure of sentences
  - semantics: truth of sentences wrt models
  - entailment: necessary truth of one sentence given another
  - inference: deriving sentences from other sentences
  - soundness: derivations produce only entailed sentences
  - completeness: derivations can produce all entailed sentences.
- The Logic Machine in Isaac Asimov's Foundation Series.