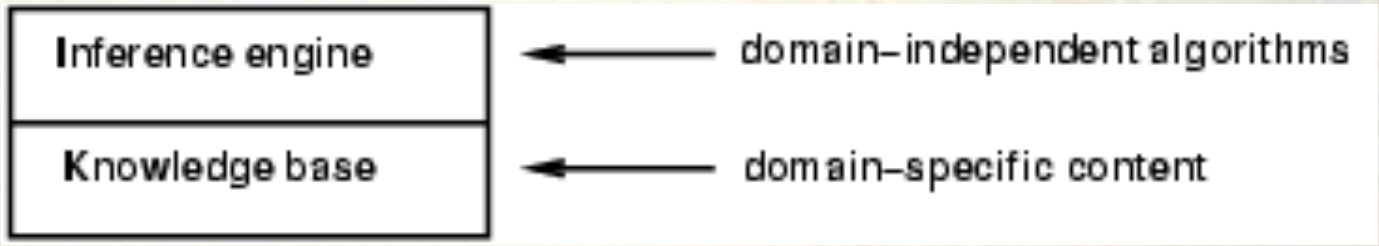
Artificial Intelligence

Logical Agent

Outline

- Knowledge-based agents
- Wumpus world
- Logic in general - models and entailment
- Propositional (Boolean) logic (logic mệnh đề)
- Equivalence (tương đương), validity (vững chắc), satisfiability (thỏa được).
- Inference rules (luật suy diễn) and theorem proving
  - resolution

Knowledge bases



- 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
- Then it can `Ask` itself what to do - answers should follow from the KB

A simple knowledge-based agent

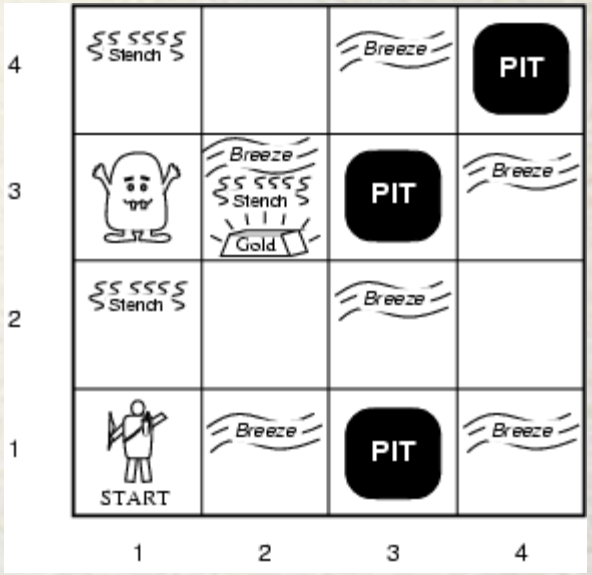
```
function KB-AGENT(percept) returns an action
    static: KB, a knowledge base
            t, a counter, initially 0, indicating time

    TELL(KB, MAKE-PERCEPT-SENTENCE(percept, t))
    action ← ASK(KB, MAKE-ACTION-QUERY(t))
    TELL(KB, MAKE-ACTION-SENTENCE(action, t))
    t ← t + 1
    return action
```

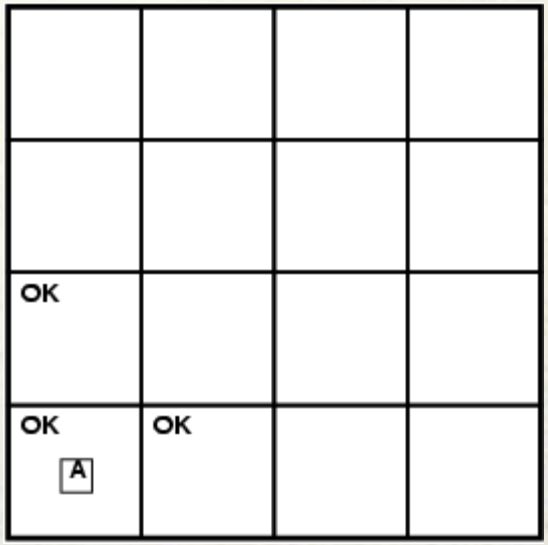
- The agent must be able to:
  - Represent states, actions, etc.
  - Incorporate new percepts
  - Update internal representations of the world
  - Deduce hidden properties of the world
  - Deduce appropriate actions

Wumpus World

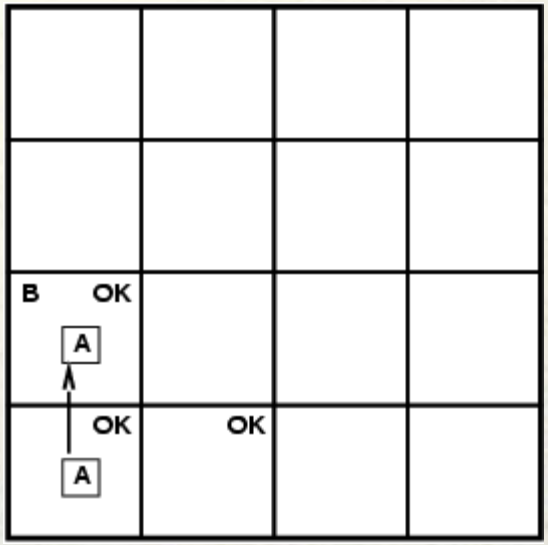
- Performance measure
  - gold +1000, death -1000
  - -1 per step, -10 for using the arrow
- Environment
  - Squares adjacent to wumpus are smelly
  - Squares adjacent to pit are breezy
  - Glitter iff gold is in the same square
  - Shooting kills wumpus if you are facing it
  - Shooting uses up the 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



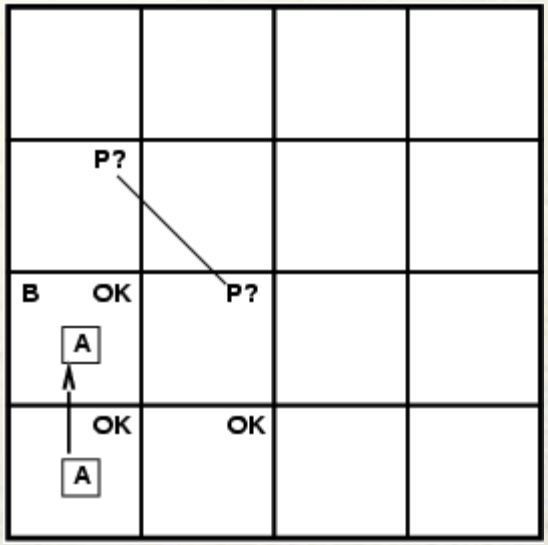
Exploring a wumpus world



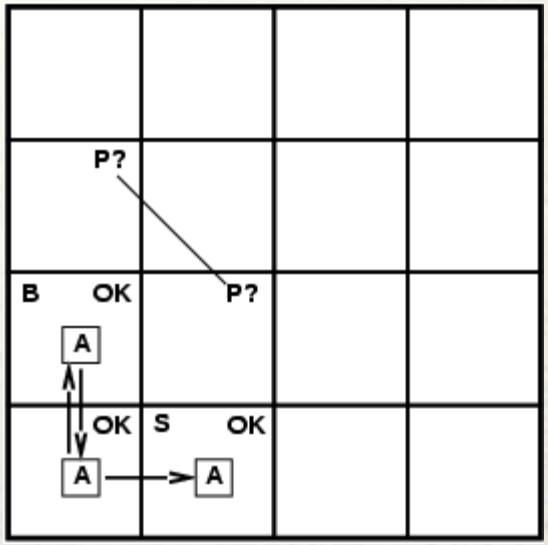
Exploring a wumpus world



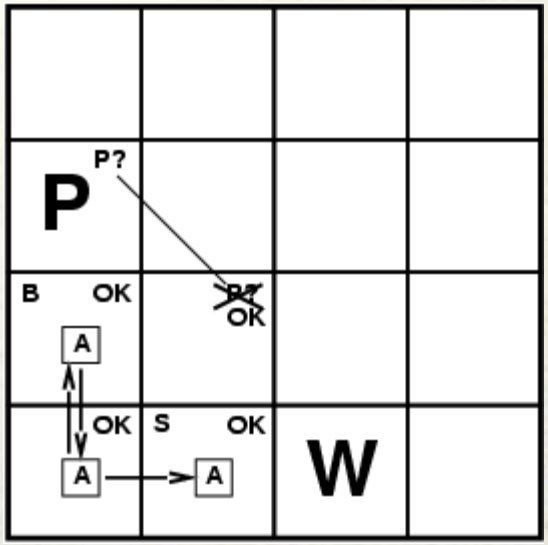
Exploring a wumpus world



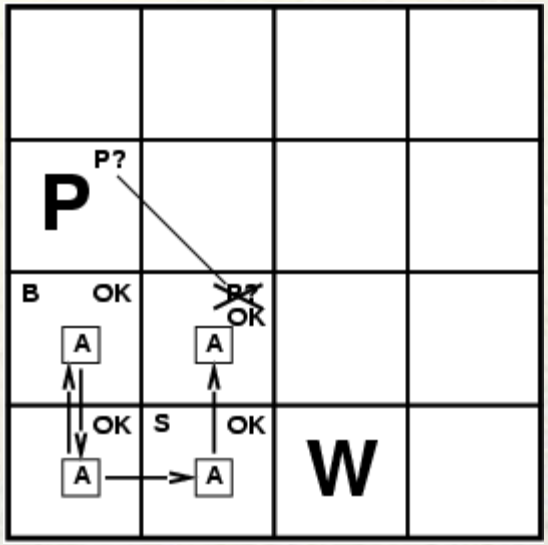
Exploring a wumpus world



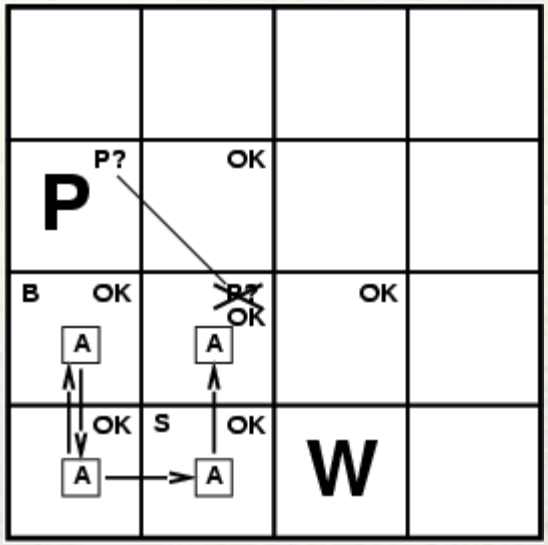
Exploring a wumpus world



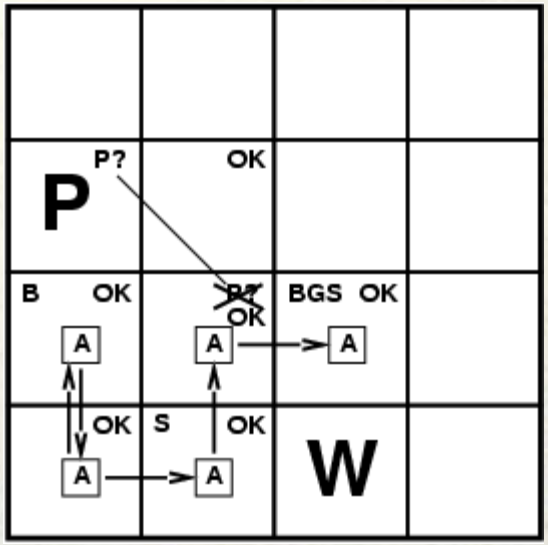
Exploring a wumpus world



Exploring a wumpus world



Exploring a wumpus world



Logic in general

- 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 iff 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

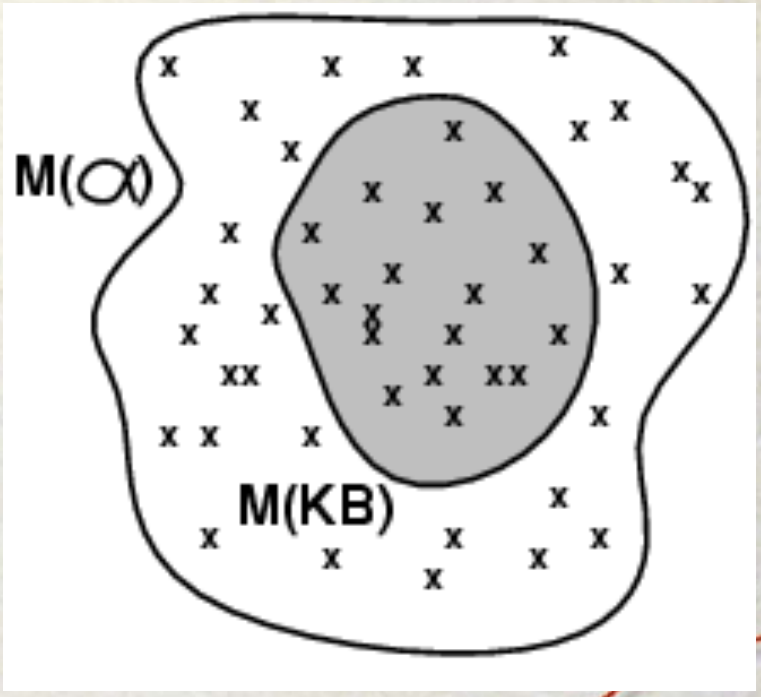
- Entailment (hệ quả) means that one thing follows from another:

$$KB \models \alpha$$

- 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

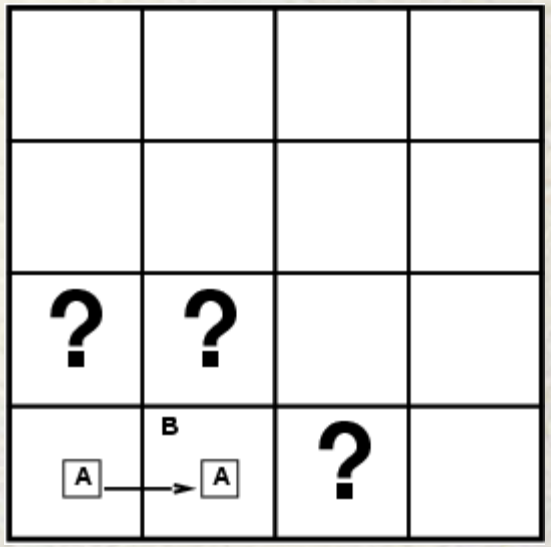
Models

- Logicians typically think in terms of models (mô hình), 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 ⊨ α iff *M(KB)* ⊆ *M(*α)
  - E.g. *KB* = Giants won and Reds won α = Giants won



Entailment in the wumpus world

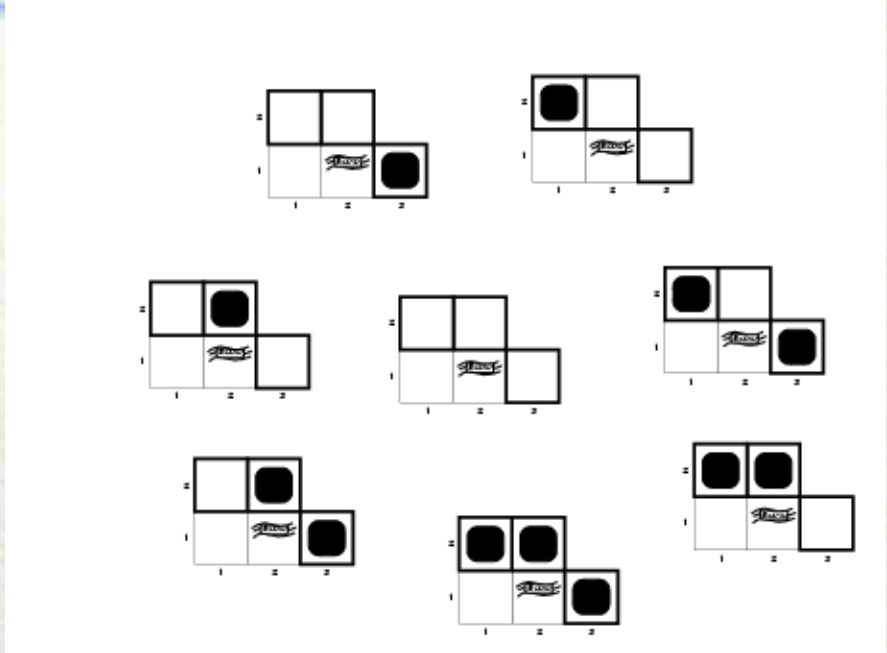
Situation after detecting nothing in [1,1], moving right, breeze in [2,1]



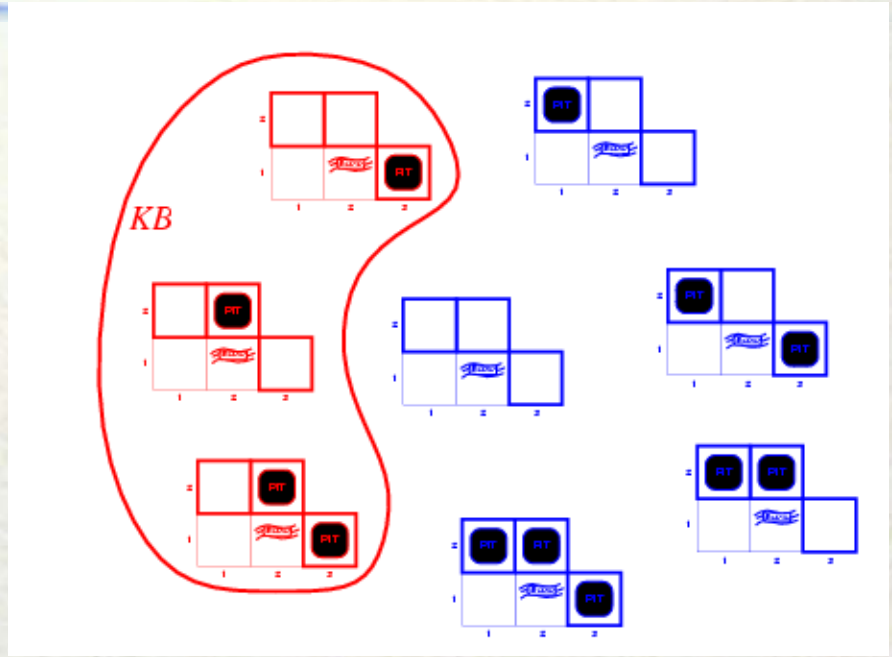
Consider possible models for *KB* assuming only pits

3 Boolean choices ⇒ 8 possible models

Wumpus models

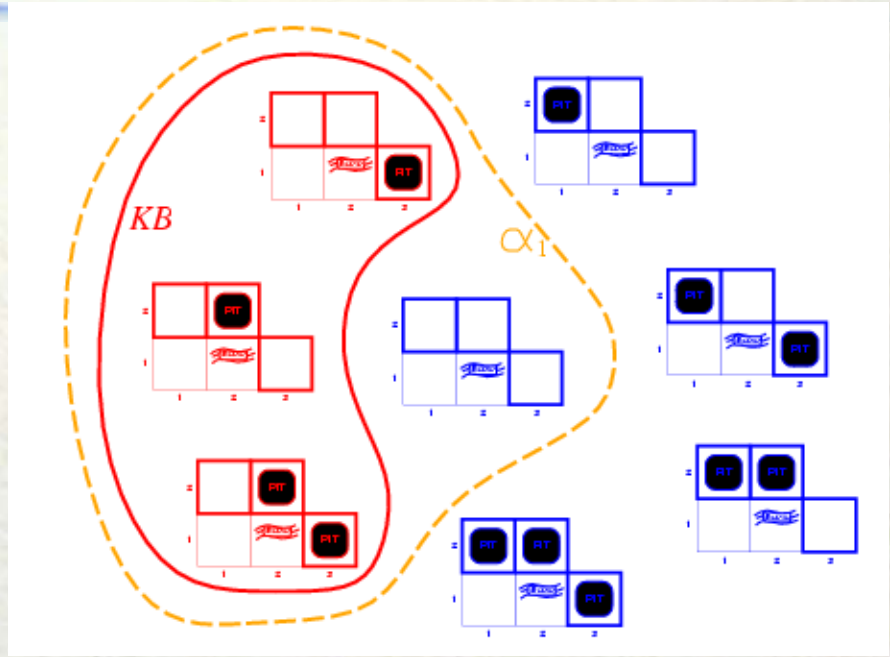


Wumpus models



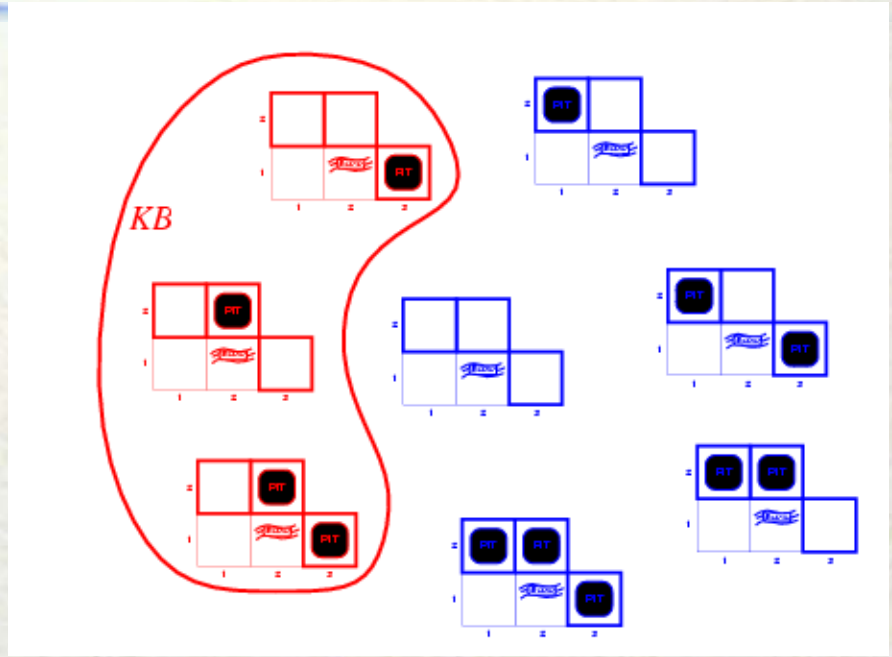
- *KB* = wumpus-world rules + observations

Wumpus models



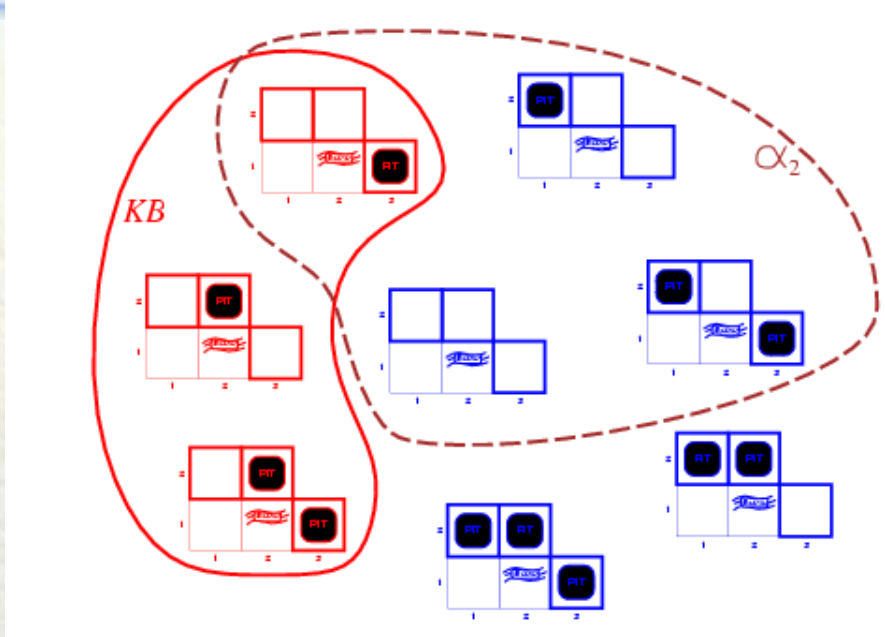
- *KB* = wumpus-world rules + observations
- α_1 = "[1,2] is safe", $KB \models \alpha_1$, proved by model checking

Wumpus models



- *KB* = wumpus-world rules + observations

Wumpus models



- *KB* = wumpus-world rules + observations
- α_2 = "[2,2] is safe", $KB \not\models \alpha_2$

Inference
Suy diễn

- $KB \vdash_i \alpha$ = sentence α can be derived from *KB* by procedure *i*
- Soundness (tin cậy): *i* is sound if whenever $KB \vdash_i \alpha$, it is also true that $KB \models \alpha$
- Completeness (đầy đủ): *i* is complete if whenever $KB \models \alpha$, it is also true that $KB \vdash_i \alpha$

Propositional logic: Syntax (Cú pháp)

- Propositional logic is the simplest logic – illustrates basic ideas
- Gồm tập các ký hiệu và tập các luật xây dựng công thức
- The proposition symbols P_1, P_2 etc are 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–hội)
  - If S_1 and S_2 are sentences, $S_1 \vee S_2$ is a sentence (disjunction-tuyển)
  - If S_1 and S_2 are sentences, $S_1 \Rightarrow S_2$ is a sentence (implication-kéo theo)
  - If S_1 and S_2 are sentences, $S_1 \Leftrightarrow S_2$ is a sentence (biconditional-kéo theo nhau)

Propositional logic: Semantics (ngữ nghĩa)

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.,

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

Ngữ nghĩa

- Bằng cách kết hợp mỗi ký hiệu mệnh đề với một mệnh đề phát biểu một khẳng định nào đó về thế giới thực
- Ví dụ: ký hiệu mệnh đề P có thể ứng với mệnh đề "Hà nội có 100 phố", khi đó P có thể nhận giá trị TRUE hoặc FALSE

Ngữ nghĩa (...)

- Ví dụ:

  P = có xe máy ở nhà

  Q = có xe buýt

  $P \vee Q$ là một công thức, nhận giá trị TRUE khi P hoặc Q nhận giá trị TRUE

Biến đổi tiếng Việt sang logic

- Hôm nay là thứ ba và trời âm u.
  - B ∧ A: B có nghĩa “Hôm nay là thứ ba” và A có nghĩa “Hôm nay trời âm u”.
  - Mức độ chi tiết khác nhau
- Câu phức tạp hơn đòi hỏi predicate(vị từ):
  - Trời âm u ở Hà Nội
  - A(H) hoặc H(A)

Truth tables for connectives

- Một minh họa là cách gán cho mỗi ký hiệu mệnh đề một giá trị chân lý. Ý nghĩa của một công thức được xác định bởi ý nghĩa của các kết nối logic.
- Bảng chân lý liệt kê tất cả các minh họa và cho phép ta xác định ngữ nghĩa của một công thức.

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 table

- Bảng chân lý cho công thức phức tạp:

A	B	C	$A \wedge (B \vee C)$
false	false	false	false
false	false	true	false
false	true	false	false
false	true	true	false
true	false	false	false
true	false	true	true
true	true	false	true
true	true	true	true

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].

$\neg P_{1,1}$

$\neg B_{1,1}$

$B_{2,1}$

- "Pits cause breezes in adjacent squares"

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

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

- "A square is breezy if and only if there is an adjacent pit"

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

Proof methods

- Proof methods divide into (roughly) two kinds:
  - 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
    - Typically require transformation of sentences into a normal form
  - 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

Normal Form
Dạng chuẩn tắc

- Xem xét việc chuẩn hóa các công thức, đưa các công thức về dạng thuận lợi hơn cho việc lập luận và suy diễn.
- Sự tương đương của các công thức
- Đưa về dạng chuẩn tắc

Sự tương đương của các công thức

- Hai công thức A và B được xem là tương đương nếu chúng có cùng một giá trị chân lý trong mọi minh họa:
- Để chỉ A tương đương với B ta viết $A \equiv B$
  - $A \wedge B \equiv B \wedge A$

Sự tương đương của các công thức (...)

- Bằng phương pháp bảng chân lý, dễ dàng chứng minh được sự tương đương của các công thức:

$$A \Rightarrow B \equiv \neg A \vee B$$

$$A \Leftrightarrow B \equiv (A \Rightarrow B) \wedge (B \Rightarrow A)$$

$$\neg (\neg A) \equiv A$$

Sự tương đương của các công thức (...)

- Luật De Morgan:

  $\neg (A \vee B) \equiv \neg A \wedge \neg B$

  $\neg (A \wedge B) \equiv \neg A \vee \neg B$

- Luật giao hoán:

  $A \vee B \equiv B \vee A$

  $A \wedge B \equiv B \wedge A$

Sự tương đương của các công thức (…)

- Luật kết hợp:

  $(A \vee B) \vee C \equiv A \vee (B \vee C)$

  $(A \wedge B) \wedge C \equiv A \wedge (B \wedge C)$

- Luật phân phối:

  $A \wedge (B \vee C) \equiv (A \wedge B) \vee (A \wedge C)$

  $A \vee (B \wedge C) \equiv (A \vee B) \wedge (A \vee C)$

Logical equivalence

- Two sentences are logically equivalent iff true in same models: α ≡ ß iff α ⊨ β and β ⊨ α

$$
\begin{aligned}
(\alpha \wedge \beta) &\equiv (\beta \wedge \alpha) && \text{commutativity of } \wedge \\
(\alpha \vee \beta) &\equiv (\beta \vee \alpha) && \text{commutativity of } \vee \\
((\alpha \wedge \beta) \wedge \gamma) &\equiv (\alpha \wedge (\beta \wedge \gamma)) && \text{associativity of } \wedge \\
((\alpha \vee \beta) \vee \gamma) &\equiv (\alpha \vee (\beta \vee \gamma)) && \text{associativity of } \vee \\
\neg(\neg\alpha) &\equiv \alpha && \text{double-negation elimination} \\
(\alpha \Rightarrow \beta) &\equiv (\neg\beta \Rightarrow \neg\alpha) && \text{contraposition} \\
(\alpha \Rightarrow \beta) &\equiv (\neg\alpha \vee \beta) && \text{implication elimination} \\
(\alpha \Leftrightarrow \beta) &\equiv ((\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)) && \text{biconditional elimination} \\
\neg(\alpha \wedge \beta) &\equiv (\neg\alpha \vee \neg\beta) && \text{de Morgan} \\
\neg(\alpha \vee \beta) &\equiv (\neg\alpha \wedge \neg\beta) && \text{de Morgan} \\
(\alpha \wedge (\beta \vee \gamma)) &\equiv ((\alpha \wedge \beta) \vee (\alpha \wedge \gamma)) && \text{distributivity of } \wedge \text{ over } \vee \\
(\alpha \vee (\beta \wedge \gamma)) &\equiv ((\alpha \vee \beta) \wedge (\alpha \vee \gamma)) && \text{distributivity of } \vee \text{ over } \wedge
\end{aligned}
$$

Dạng chuẩn tắc (CNF)

- Các công thức tương đương có thể xem như các biểu diễn khác nhau của cùng một sự kiện
- Để dễ dàng viết các chương trình máy tính thao tác trên các công thức, chúng ta sẽ chuẩn hóa các công thức – đưa về dạng chuẩn hội - conjunctive normal form (CNF)
- Dạng chuẩn hội nếu là hội ($\wedge$) của các câu tuyển ($\vee$)
- Câu tuyển có dạng $A_1 \vee \ldots \vee A_m$, trong đó A_i là ký hiệu mệnh đề (P) hoặc phủ định của ký hiệu mệnh đề ($\neg$ P)

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 distributivity law ($\wedge$ over $\vee$) and flatten:
   $(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge (\neg P_{1,2} \vee B_{1,1}) \wedge (\neg P_{2,1} \vee B_{1,1})$

Dạng chuẩn tắc (…)

- Chúng ta có thể biến đổi một công thức bất kỳ về công thức ở dạng chuẩn hội bằng cách áp dụng thủ tục sau:
  - Bỏ các dấu kéo theo ⇒ bằng cách thay A ⇒ B bởi ¬ A ∨ B
  - Chuyển các dấu phủ định ¬ vào sát các ký hiệu mệnh đề bằng cách áp dụng luật De Morgan và thay ¬ (¬ A) bởi A
  - Áp dụng luật phân phối, thay các công thức có dạng A ∨ (B ∧ C) bởi (A ∨ B) ∧ (A ∨ C)

Dạng chuẩn tắc (…)

Ví dụ: $(P \Rightarrow Q) \vee \neg (R \vee \neg S)$

$(P \Rightarrow Q) \vee \neg (R \vee \neg S) \equiv (\neg P \vee Q) \vee (\neg R \wedge S)$

$\equiv ((\neg P \vee Q) \vee \neg R) \wedge ((\neg P \vee Q) \vee S)$

$\equiv (\neg P \vee Q \vee \neg R) \wedge (\neg P \vee Q \vee S)$

Như vậy công thức $(P \Rightarrow Q) \vee \neg (R \vee \neg S)$ được đưa về dạng chuẩn hội $(\neg P \vee Q \vee \neg R) \wedge (\neg P \vee Q \vee S)$

Luật suy diễn

- Một công thức H được xem là hệ quả logic của một tập công thức G = {G_1,…,G_m} nếu từ G_1, … , G_m ta có thể suy ra H
- Khi có một cơ sở tri thức, ta muốn sử dụng các tri thức trong cơ sở này để suy ra tri thức mới mà nó là hệ quả logic của các công thức trong cơ sở tri thức.
  - Được thực hiện bằng cách sử dụng các luật suy diễn

Inference Rules

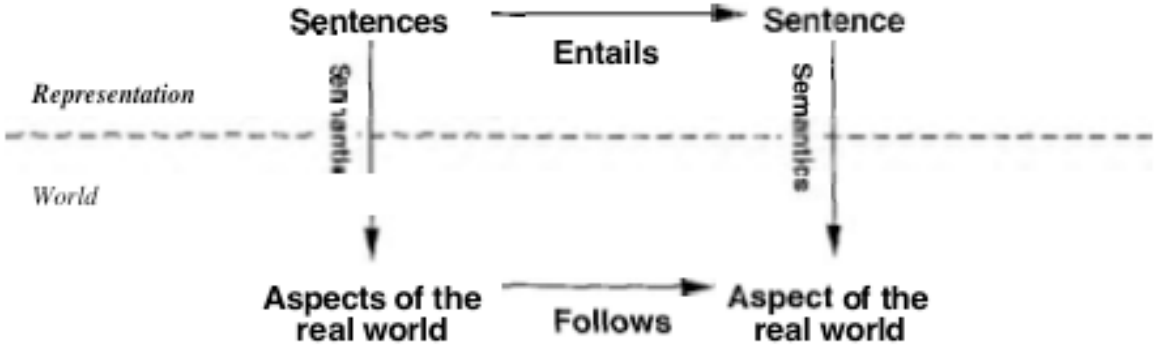


Figure 7.6 Sentences are physical configurations of the agent, and reasoning is a process of constructing new physical configurations from old ones. Logical reasoning should ensure that the new configurations represent aspects of the world that actually follow from the aspects that the old configurations represent.

Luật suy diễn

- Một luật suy diễn gồm hai phần: một tập các điều kiện (tiên đề) và một kết luận (định lý)
- Ta sẽ biểu diễn các luật suy diễn dưới dạng "phân số":
  - trong đó tử số là danh sách các điều kiện
  - còn mẫu số là kết luận của luật
  - mẫu số là công thức mới được suy ra từ các công thức ở tử số

Luật suy diễn (…)

Một số luật suy diễn quan trọng trong logic mệnh đề:

- Luật Modus Ponens
- Luật Modus Tollens
- Luật bắc cầu
- Luật loại bỏ hội
- Luật đưa vào hội
- Luật đưa vào tuyển
- Luật phân giải
- Luật Reductio Ad Absurdum
- Luật đưa vào kéo theo
- Luật loại bỏ phủ định

Luật Modus Ponens

$$\frac{\alpha \Rightarrow \beta,\ \alpha}{\beta}$$

Luật Modus Tollens

$$\alpha \Rightarrow \beta, \neg \beta$$

$$\neg \alpha$$

Luật bắc cầu

$$\alpha \Rightarrow \beta,\ \beta \Rightarrow \gamma$$

$$\alpha \Rightarrow \gamma$$

Luật loại bỏ hội

$$\frac{\alpha_1 \wedge \ldots \wedge \alpha_m}{\alpha_i}$$

Luật đưa vào hội

$$\frac{\alpha_1, \ldots, \alpha_m}{\alpha_1 \wedge \ldots \wedge \alpha_m}$$

Luật đưa vào tuyển

$$\frac{\alpha_i}{\alpha_1 \vee \ldots \vee \alpha_m}$$

Luật phân giải

$$\frac{\alpha \vee \beta,\ \neg\beta \vee \gamma}{\alpha \vee \gamma}$$

Resolution
Luật phân giải

Conjunctive Normal Form (CNF)

conjunction of disjunctions of literals clauses

E.g., $(A \vee \neg B) \wedge (B \vee \neg C \vee \neg D)$

- Resolution inference rule (for CNF):

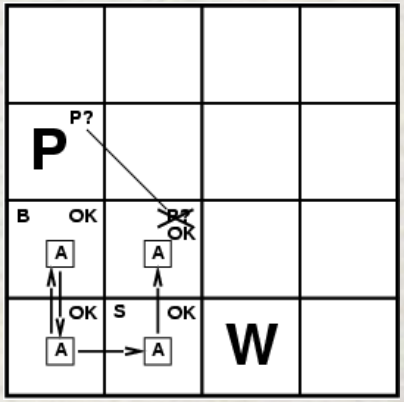
$$\frac{\ell_i \vee \ldots \vee \ell_k, \qquad m_1 \vee \ldots \vee m_n}{\ell_i \vee \ldots \vee \ell_{i-1} \vee \ell_{i+1} \vee \ldots \vee \ell_k \vee m_1 \vee \ldots \vee m_{j-1} \vee m_{j+1} \vee \ldots \vee m_n}$$

where ℓ_i and m_j are complementary literals.

E.g., $P_{1,3} \vee P_{2,2}, \qquad \neg P_{2,2}$

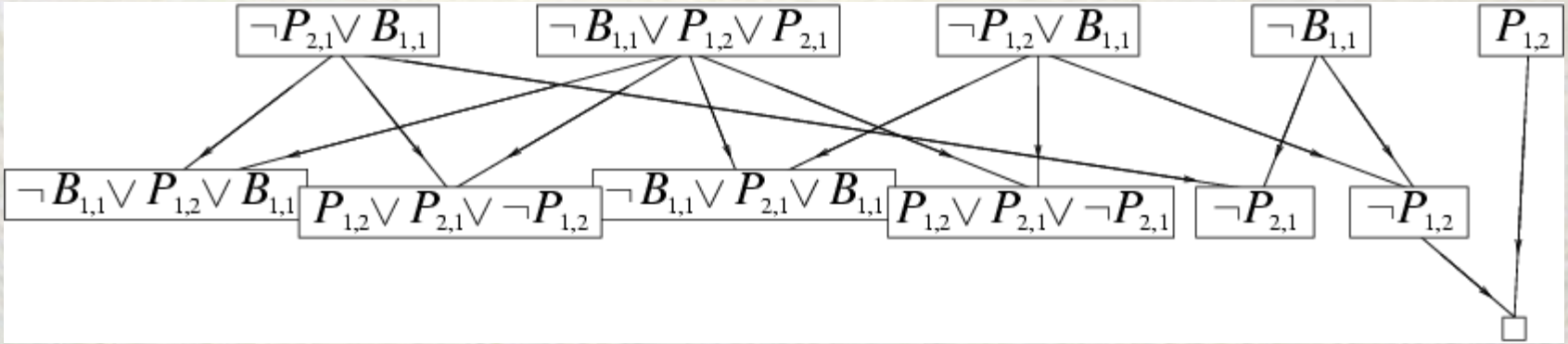
$$\frac{}{P_{1,3}}$$

- Resolution is sound and complete for propositional logic



Resolution example

- $KB = (B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})) \wedge \neg B_{1,1}$
- $\alpha = \neg P_{1,2}$



Luật phân giải (…)

- Chứng minh luật Modus Ponens, Modus Tollens và luật bắc cầu là trường hợp riêng của luật phân giải

Luật Reductio Ad Absurdum

$$\frac{\begin{array}{c}\neg\,\alpha \\ . \\ . \\ \bot\end{array}}{\alpha}$$

- Sau đó phải loại bỏ giả thiết ¬ α
- Ký hiệu ⊥ là: biểu tượng mâu thuẫn (có thể ký hiệu là □ là câu rỗng khi áp dụng luật phân giải) :

$$\frac{\neg\,\beta,\ \beta}{\bot}$$

Luật đưa vào kéo theo

α

.

.

β

α ⇒ β

- Sau đó phải loại bỏ giả thiết α

Luật loại bỏ phủ định

$$\frac{\neg\neg\alpha}{\alpha}$$

Định lý suy diễn
Deduction Theorem

- S là một tập các công thức, A, B là 2 công thức
- Nếu từ $S \cup \{A\}$ suy ra B thì từ S suy ra $(A \Rightarrow B)$
- Nếu từ S suy ra $(A \Rightarrow B)$ thì từ $S \cup \{A\}$ suy ra B
- Ví dụ: chứng minh
  $\{A \Rightarrow B\} \vdash A \Rightarrow (C \Rightarrow B)$

Chứng minh bác bỏ bằng luật phân giải

- Ý tưởng:

Từ G muốn chứng minh A, ta chứng minh từ G và ¬ A sẽ suy ra được câu rỗng (hay mâu thuẫn)

Chứng minh bằng luật phân giải (...)

- Thủ tục phân giải

Đầu vào: Tập G các câu tuyển

Begin

1. **Repeat**

   1.1 Chọn hai câu A và B thuộc G

   1.2 **if** A và B phân giải được **then** áp dụng, nếu tạo ra câu mới thì thêm vào G

   Until nhận được câu rỗng hoặc không có câu mới xuất hiện

2. **If** nhận được câu rỗng **then** thông báo không thỏa được

End;

Chứng minh bằng luật phân giải (...)

- Thêm ¬ A vào G
- Chuyển hết các câu trong G và dạng chuẩn hội sau đó tách ra thành tập các câu tuyển G'
- Áp dụng thủ tục phân giải với G'
- Nếu tạo ra được câu rỗng thì đã chúng minh được, nếu không thì A không phải là hệ quả của G

Chứng minh bằng luật phân giải (...)

- Ví dụ:

Chứng minh từ ¬ A ∨ ¬ B ∨ P, ¬ C ∨ ¬ D ∨ P, ¬ E ∨ C, A, E, D chứng minh P

Expressiveness limitation of propositional logic

- KB contains "physics" sentences for every single square
- “A square is breezy if and only if there is an adjacent pit”. This statement must be converted into a separate sentence for each square.

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

- Rapid proliferation of clauses
- What we really want is a way to express such a statement in one sentence for all squares e.g.

$$\text{Breezy}(i, j) \Leftrightarrow (\text{Pit}(i-1, j) \vee \text{Pit}(i+1, j) \vee \text{Pit}(i, j-1) \vee \text{Pit}(i, j+1))$$

- First-order logic will allow us to do this.

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
- Wumpus world requires the ability to represent partial and negated information, reason by cases, etc.
- Resolution is complete for propositional logic
- Propositional logic lacks expressive power

References

- Artificial Intelligence: A modern approach. Chapter 7.
- Artificial Intelligence Illuminated. Chapter 7.