FACULTY OF INFORMATION TECHNOLOGY

Artificial Intelligence Fundamentals (NM TTNT)

Semester 1, 2021/2022

Chapter 6. Logical Agents

Content

- Knowledge-based agents
- Wumpus world
- Logic in general - models and entailment
- Propositional logic
  - Equivalence, validity, satisfiability
- Inference rules and theorem proving
  - Wang's algorithm (Wang Hao)
  - Resolution
  - Forward chaining
  - Backward chaining

A knowledge-based agent

- A knowledge-based agent includes a knowledge base and an inference system.
- **A knowledge base** is a set of representations of facts of the world.
  - Each individual representation is called a **sentence**.
  - The sentences are expressed in a **knowledge representation language**.
- **The inference mechanism**: a mechanism for determining what follows from what has been TELLed to the knowledge base. The ASK operation utilizes this inference mechanism.

A knowledge-based agent

- The agent operates as follows:
  1. It TELLs the knowledge base what it perceives.
  2. It ASKs the knowledge base what action it should perform.
  3. It performs the chosen action.

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

  TELL(KB,MAKE-PERCEPT-SENTENCE($percept, t$))
  $action \leftarrow$ ASK(KB,MAKE-ACTION-QUERY(t))
  TELL(KB,MAKE-ACTION-SENTENCE($action, t$))
  $t \leftarrow t + 1$
  return *action*

A knowledge-based agent

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

Architecture of a knowledge-based agent

- **Knowledge Level**
  - The most abstract level: describe agent by saying what it knows.
- **Logical Level**
  - The level at which the knowledge is encoded into sentences.
- **Implementation Level**
  - The physical representation of the sentences in the logical level.

Knowledge-based Agents

- Example:
  - Knowledge level:
    - The automated taxi driver knows that Golden Gate Bridge links San Francisco and Marin County.
  - Logical level:
    - The automated taxi driver has the FOL sentence Links (GGBridge, SF, Marin) in its KB.
  - Implementation level:
    - The sentence Links(GGBridge, SF, Marin) is implemented by a C structure (or a Prolog fact).

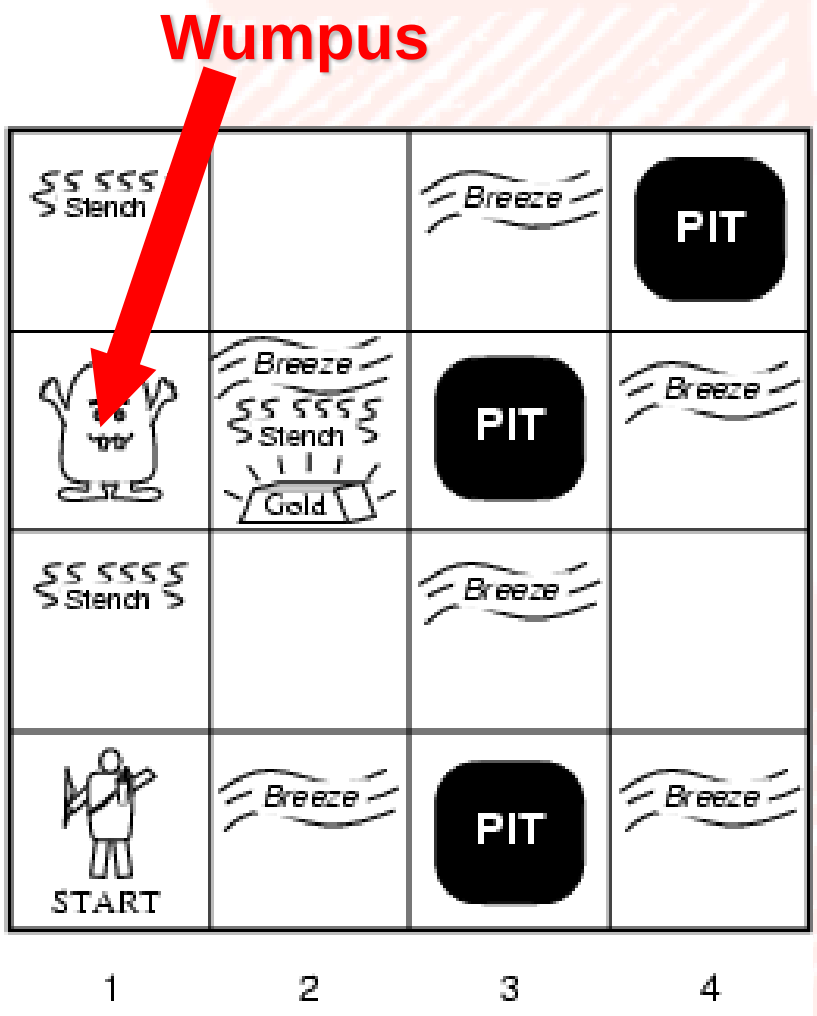
Knowledge-based Agents

- One can build a knowledge-based agent by TELLing it what it needs to know before it starts perceiving the world.

- One can also design learning mechanisms that output general knowledge about the environment given a series of percepts.

Autonomous agent = Knowledge-based agent + Learning mechanism

Knowledge-based agents: Wumpus World



- Wumpus world: a cave consisting of rooms connected by passageway.
- Lurking somewhere in the cave is the *wumpus*, a beast that eats anyone who enters its room.
- The *wumpus* can be shot by an agent, but the agent has only one arrow.
- Some rooms contain *bottomless pits* that will trap anyone who wanders into these rooms.
- The agent's goal is to find the *gold* and bring it back to the start as quickly as possible, without getting killed.

Wumpus World PEAS description

- Performance measure
  - gold +1000, death -1000
  - -1 per step, -10 for using the arrow
- Environment
  - A grid of rooms surrounded by walls can contain agent and object
  - The agent always starts in the square labeled [1, 1]
  - The locations of the gold and the wumpus are chosen randomly

Wumpus World PEAS description

- Sensors:

  (Stench, Breeze, Glitter, Bump, Scream)

  - In the squares directly (not diagonally) adjacent squares to the wumpus, the agent will perceive a stench.
  - In the squares directly adjacent to a pit, the agent will perceive a breeze.
  - In the square where the gold is, the agent will perceive a glitter.
  - When an agent walks into a wall, it will perceive a bump.
  - When the wumpus is killed, it emits a woeful scream that can be perceived anywhere in the cave.

Wumpus World PEAS description

- Actuators:
  - Left turn, Right turn
  - Forward
    - Moving forward has no effect if there is a wall in front of the agent
  - Grab
    - to pick up an object that is in the same square as the agent.
  - Shoot
    - to fire an arrow in the direction the agent is facing. The arrow continues until it either hits and kills the wumpus or hits a wall.
    - The agent only has one arrow, so only the first Shoot has effect
  - Release
    - to leave the cave; it is effective only when the agent is in the start square.

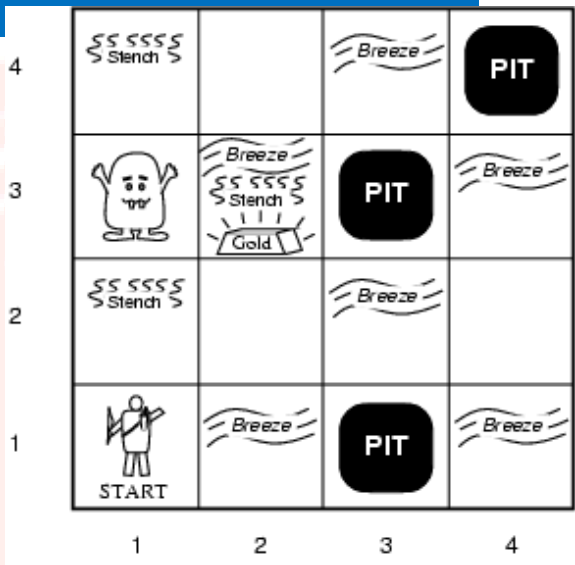
Wumpus world characterization

- Fully Observable
  - No – only local perception
- Deterministic
  - Yes – outcomes exactly specified
- 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

1, 4	2, 4	3, 4	4, 4
1, 3	2, 3	3, 3	4, 3
OK 1, 2	2, 2	3, 2	4, 2
OK A 1, 1	OK 2, 1	3, 1	4, 1

A Agent
B Breeze
G Glliter, Gold
OK Safe Square
P Pit
S Stench
V Visitted
W Wumpus



The first percept:

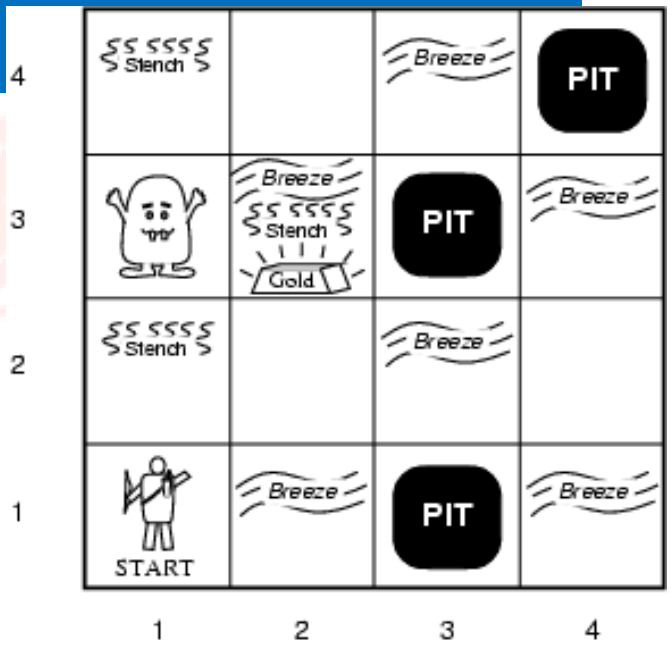
[*None, None, None, None, None*].

There was no stench or breeze in [1,1], the agent can infer that [1,2] and [2,1] are free of dangers. They are marked with an OK

Exploring a wumpus world

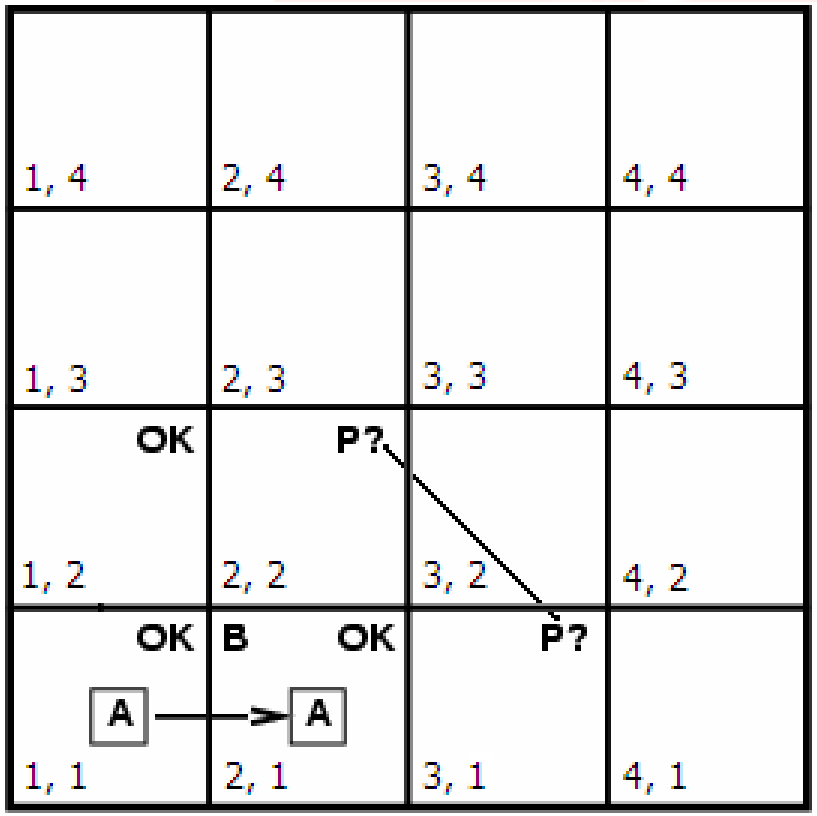
1, 4	2, 4	3, 4	4, 4
1, 3	2, 3	3, 3	4, 3
OK 1, 2	2, 2	3, 2	4, 2
OK A 1, 1	B OK A 2, 1	3, 1	4, 1

A Agent
B Breeze
G Glliter, Gold
OK Safe Square
P Pit
S Stench
V Visitted
W Wumpus

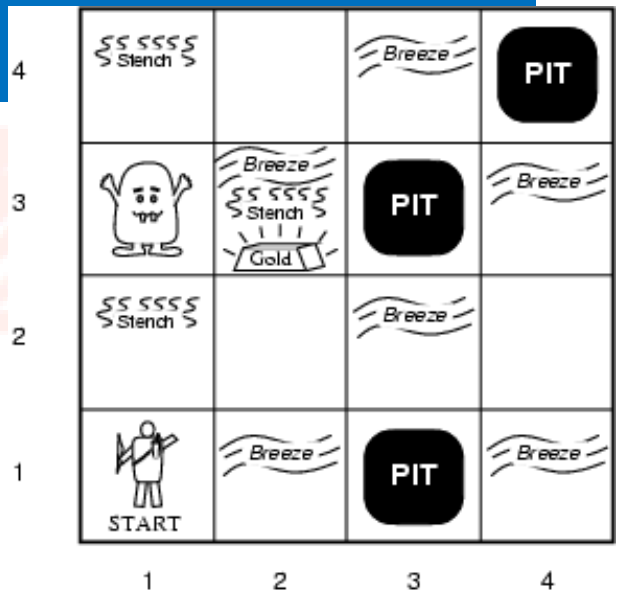


Agent decides to move forward to [2,1]. After moving, percept is [*None, Breeze, None, None, None*]

Exploring a wumpus world



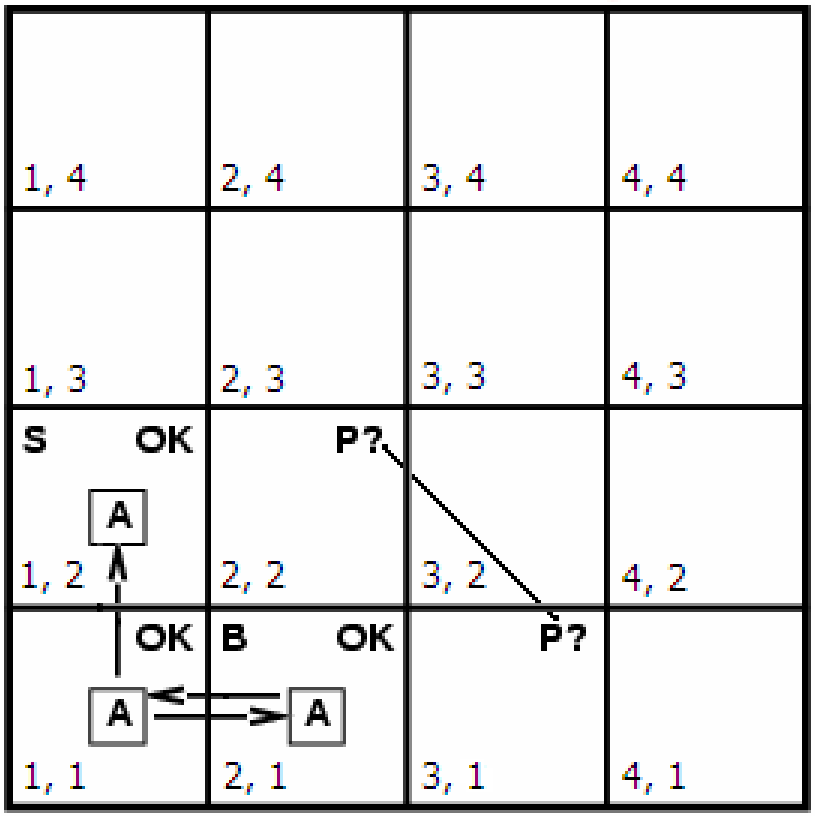
A Agent
B Breeze
G Glliter, Gold
OK Safe Square
P Pit
S Stench
V Visitted
W Wumpus



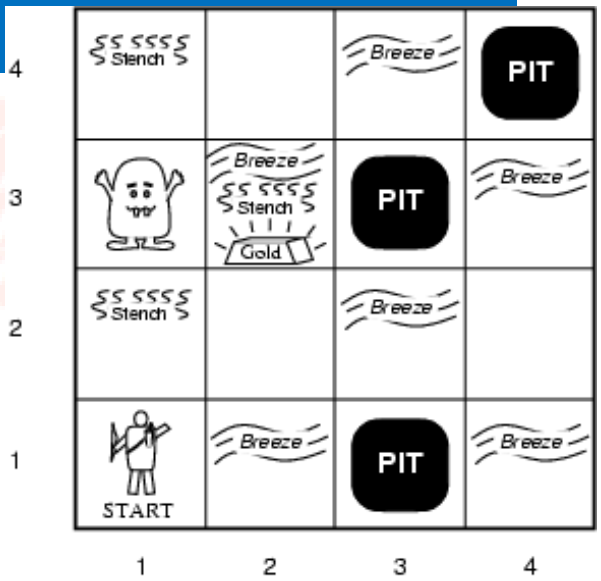
Breeze in [2,1], so there must be a pit in [2,2] or [3,1] or both.

The notation P? indicates a possible pit in those squares.

Exploring a wumpus world

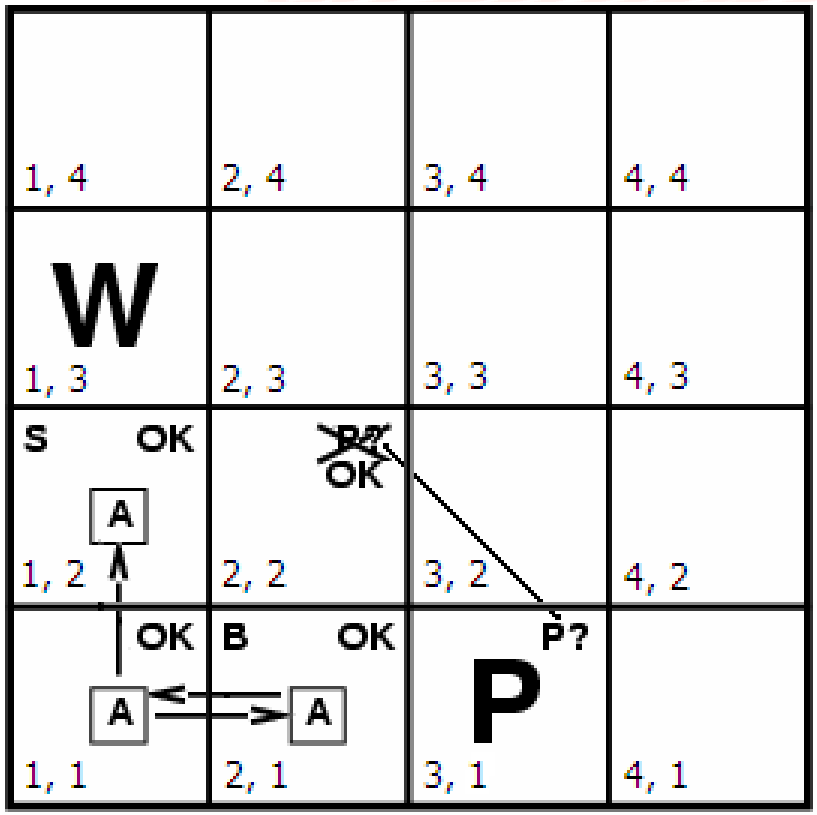


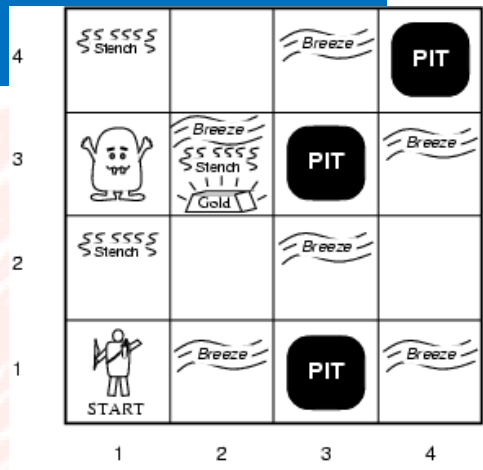
A Agent
B Breeze
G Glliter, Gold
OK Safe Square
P Pit
S Stench
V Visitted
W Wumpus



There is only one known square that is OK and has not been visited yet. So the agent will turn around, go back to [1,1], and then proceed to [1,2]. The new percept is [*Stench, None, None, None, None*].

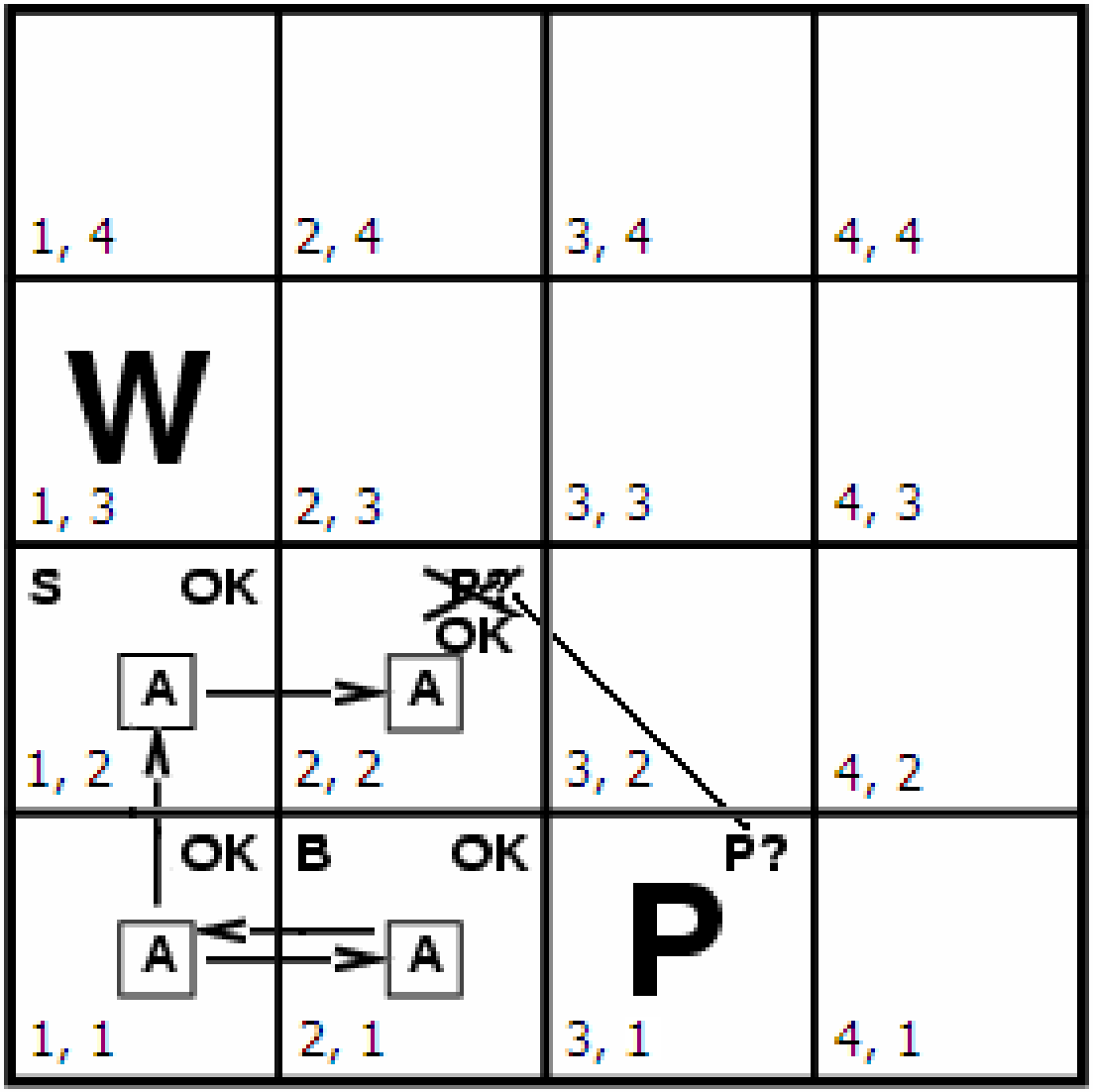
Exploring a wumpus world

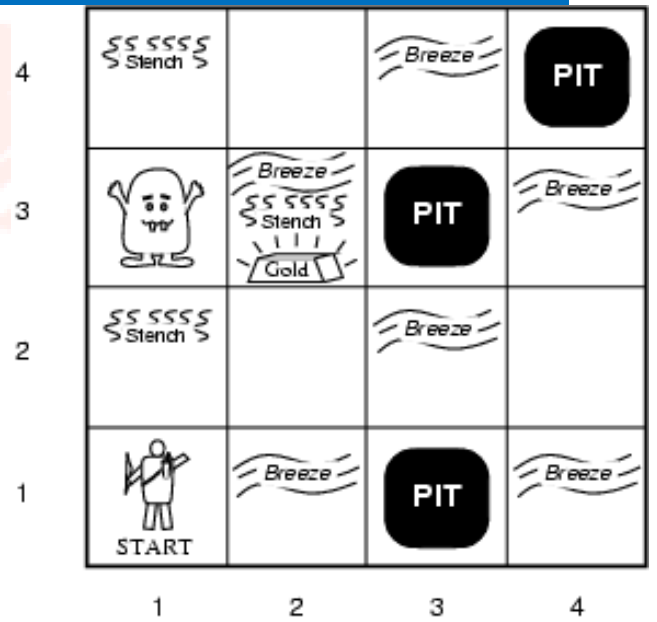




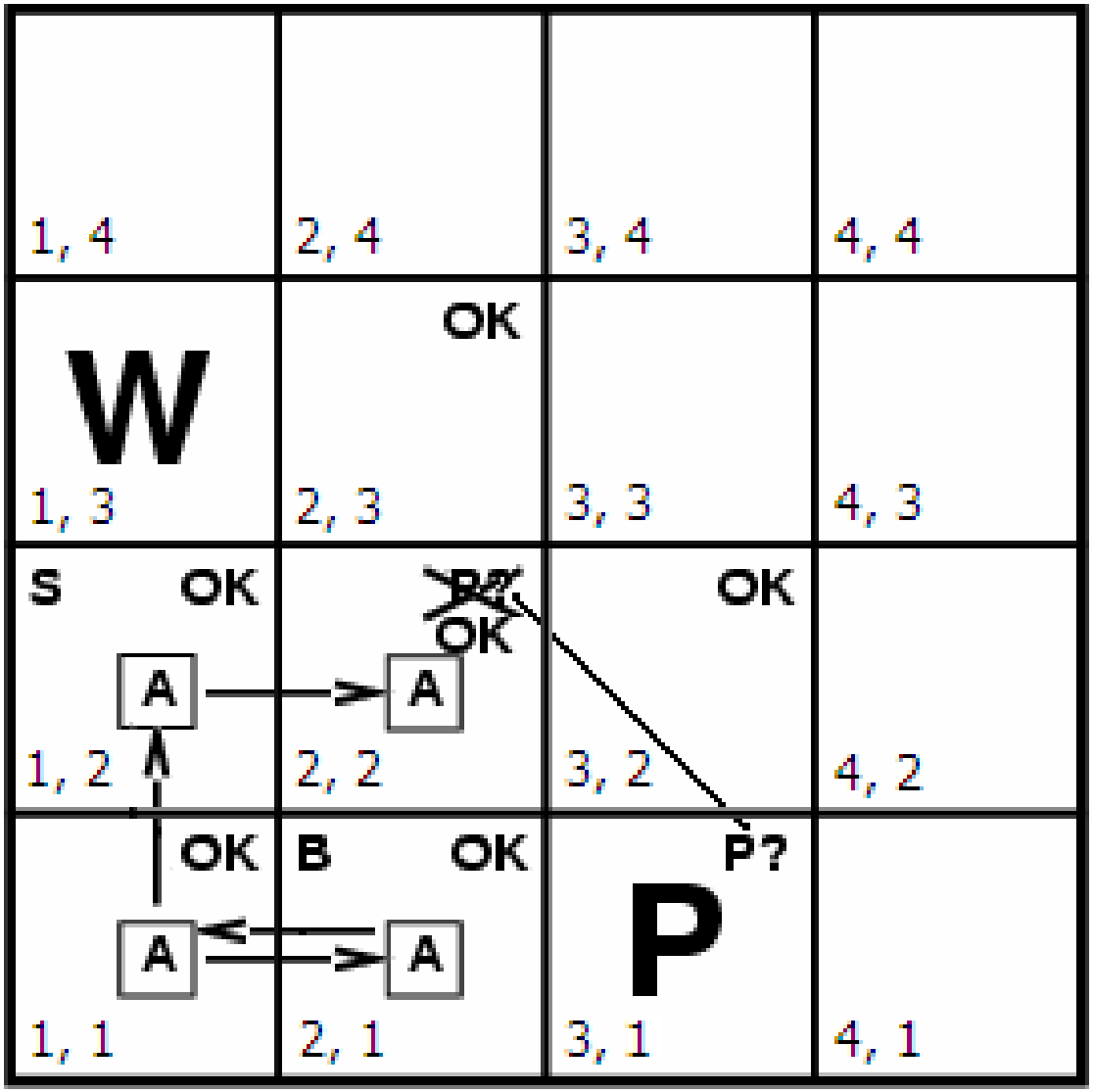
- Stench in [1,2] means that there must be a wumpus nearby.
- The wumpus can not be in [1,1] and [2,2] (or the agent would have detected a stench when it was in [2,1]). Therefore, the wumpus is in [1,3].
- Moreover, the lack of a Breeze in [1,2] implies that there is no pit in [2,2]. So pit must be in [3,1] and [2,2] is OK

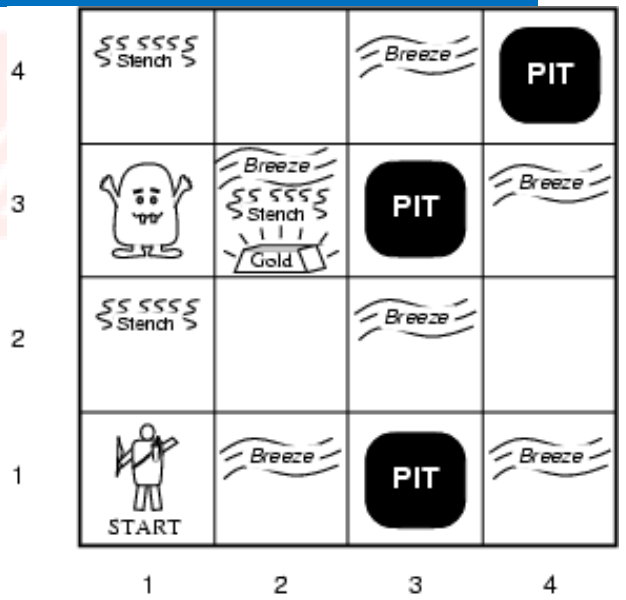
Exploring a wumpus world



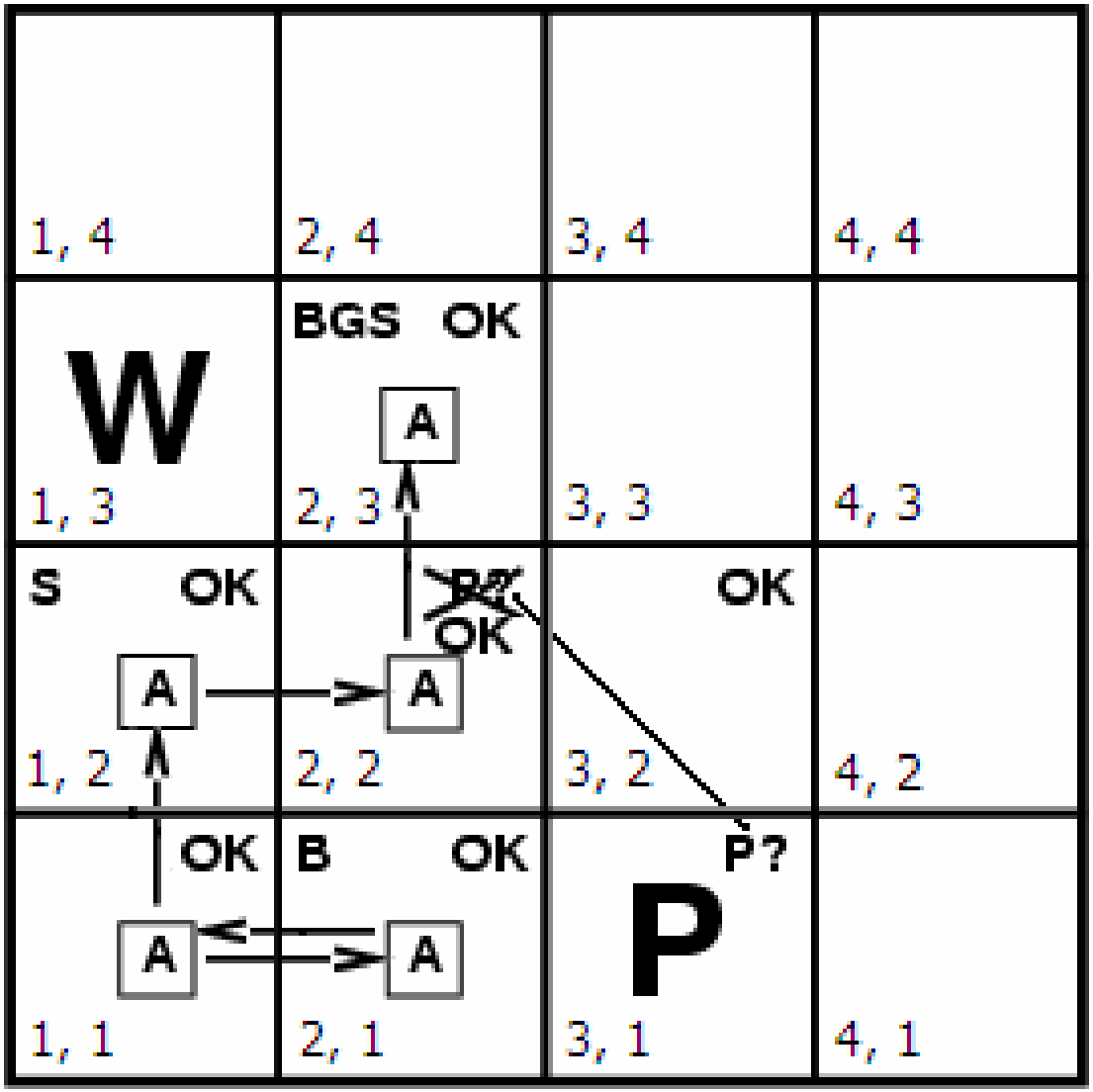


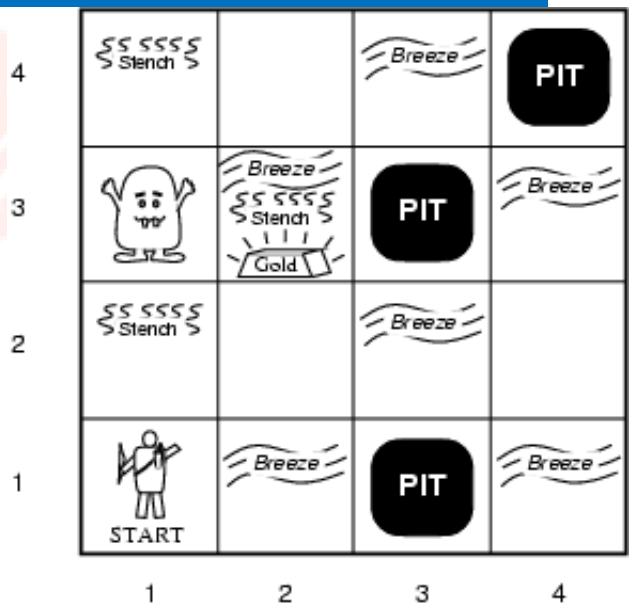
Exploring a wumpus world





Exploring a wumpus world





Logic in general

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 if 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 $x = 0; y = 6$

Characterized

- Logics are characterized by what they consider to be "primitives"

Logic	Primitives	Available Knowledge
Propositional	facts (propositions)	true/false/unknown
First-Order	facts, objects, relations	true/false/unknown
Temporal	facts, objects, relations, times	true/false/unknown
Probability Theory	facts	degree of belief 0…1
Fuzzy	degree of truth	degree of belief 0…1

Entailment – Tính bao hàm

- Entailment means that one thing follows from another:

  $KB \models \alpha$ (or $KB \Rightarrow \alpha$)

- Knowledge base KB entails sentence α if and only if α is true *in all worlds* where KB is true
  - E.g., KB: {"sky is blue", "grass is green“}
    $KB \models$ "sky is blue and grass is green"
  - E.g., x+y = 4 entails 4 = x+y

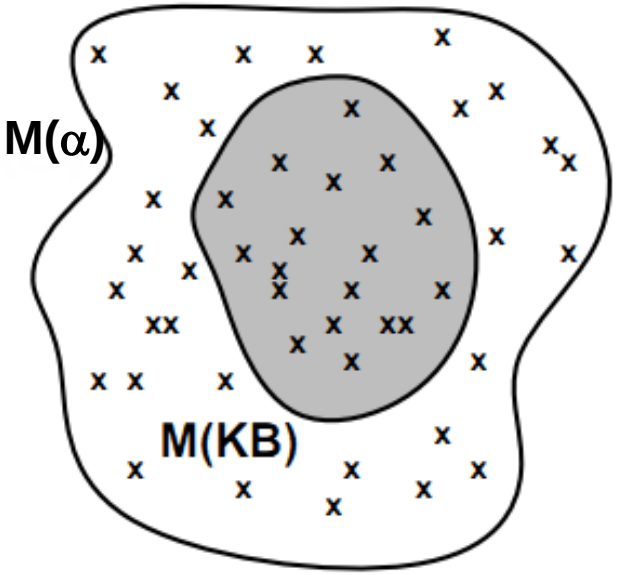
- Entailment is a relationship between sentences (i.e., syntax) that is based on semantics

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
- $M(\alpha)$ is the set of all models of α, then KB $\models \alpha$ if and only if $M(KB) \subseteq M(\alpha)$

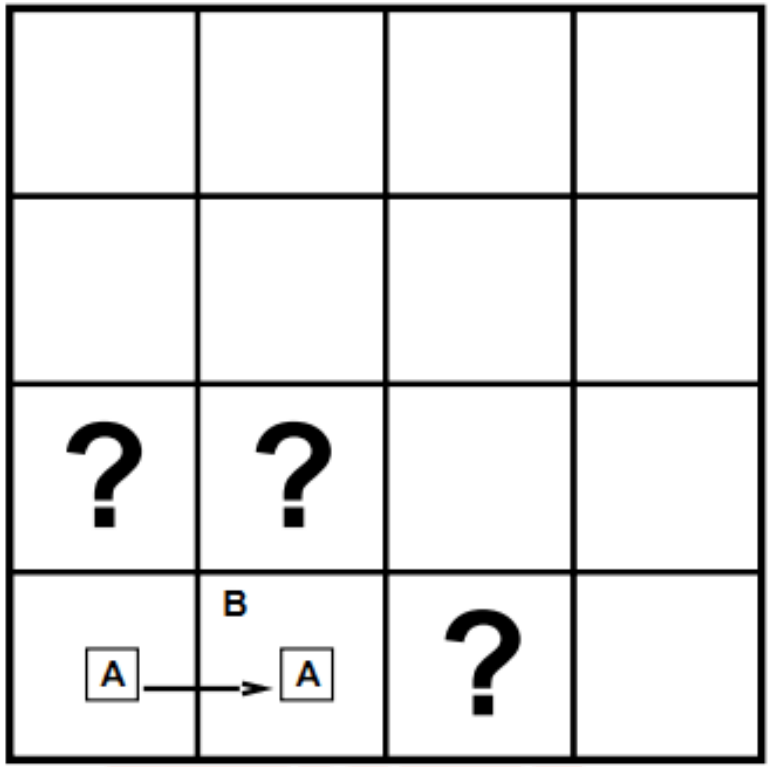
- E.g. KB = Giants won and Reds won

  α = Giants won

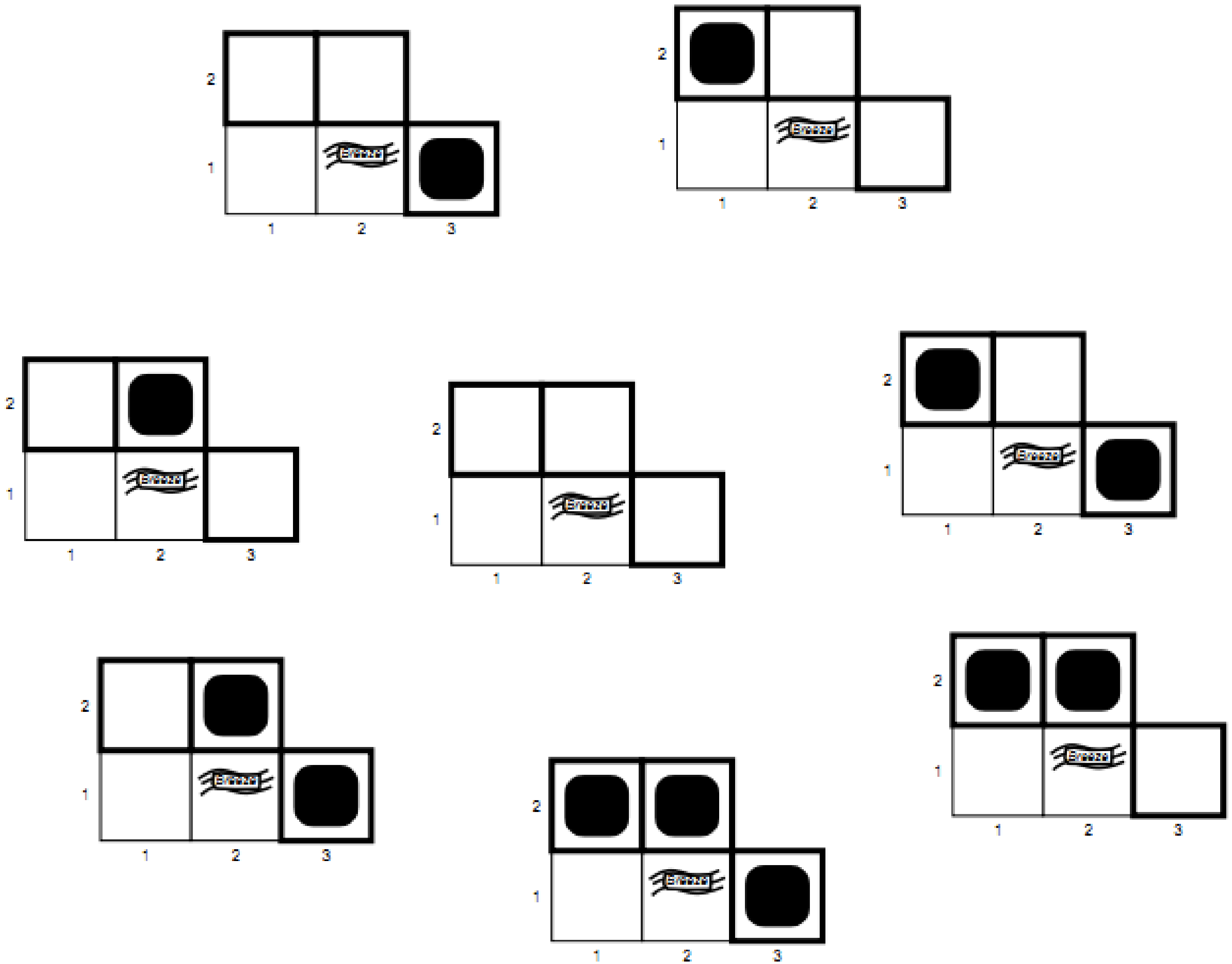


Entailment in the wumpus world

- Situation after detecting nothing in [1,1], moving right, breeze in [2,1]
- Consider possible models for? - assuming only pits
- 3 Boolean choices ➔ 8 possible models

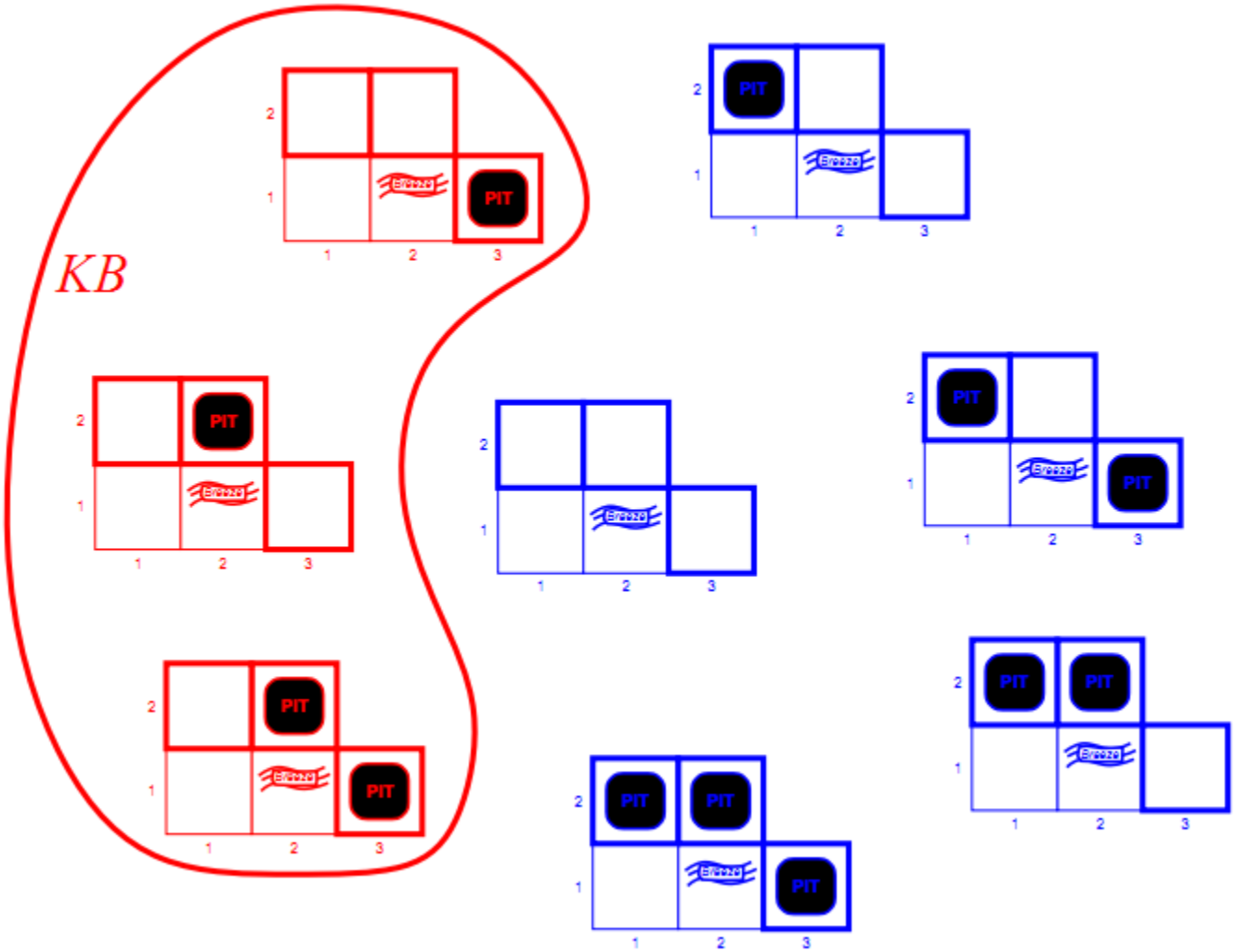


Wumpus models (cont.)



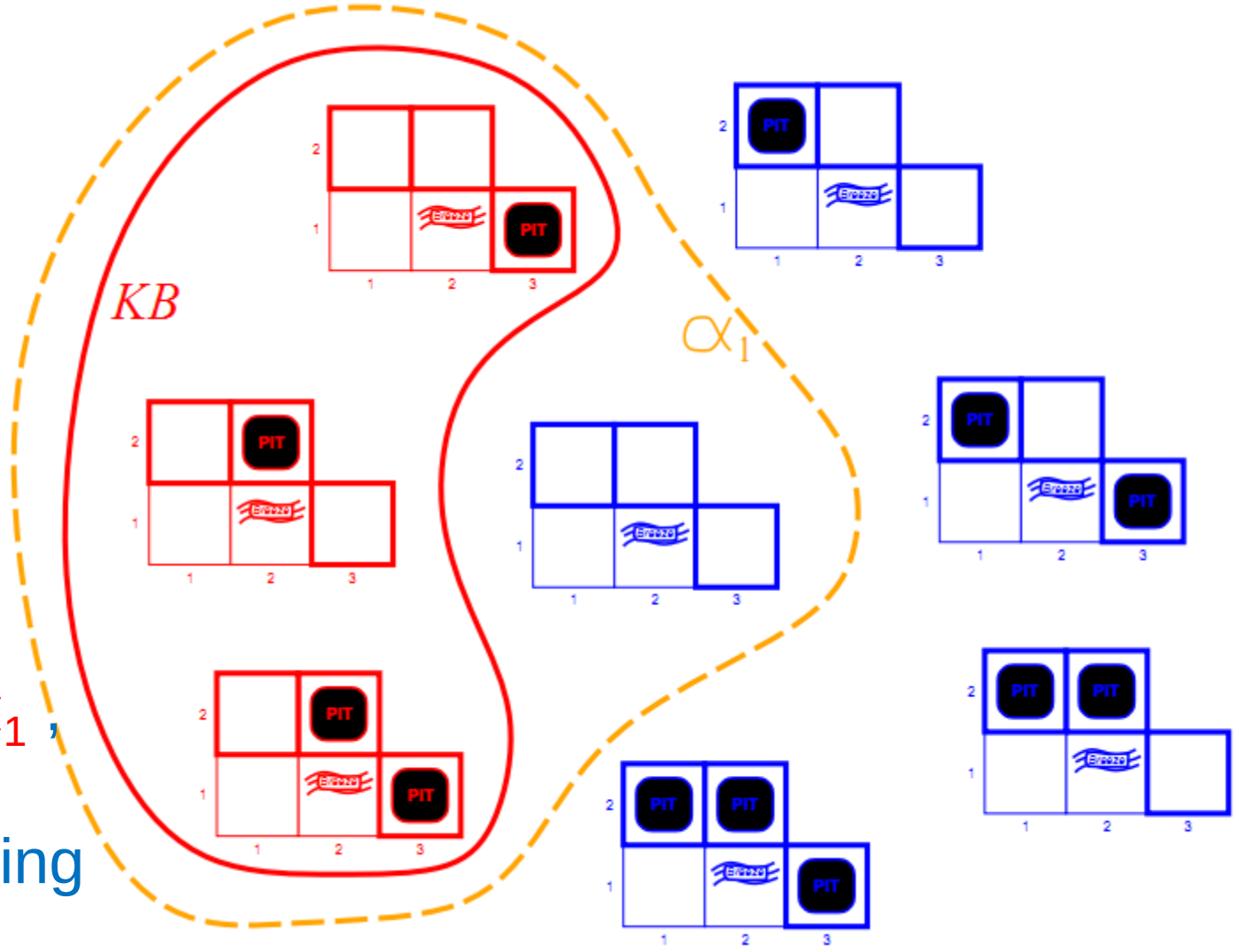
Wumpus models (cont.)

- The KB corresponding to the observations of nothing in [1,1] and a breeze in [2,1]



Wumpus models (cont.)

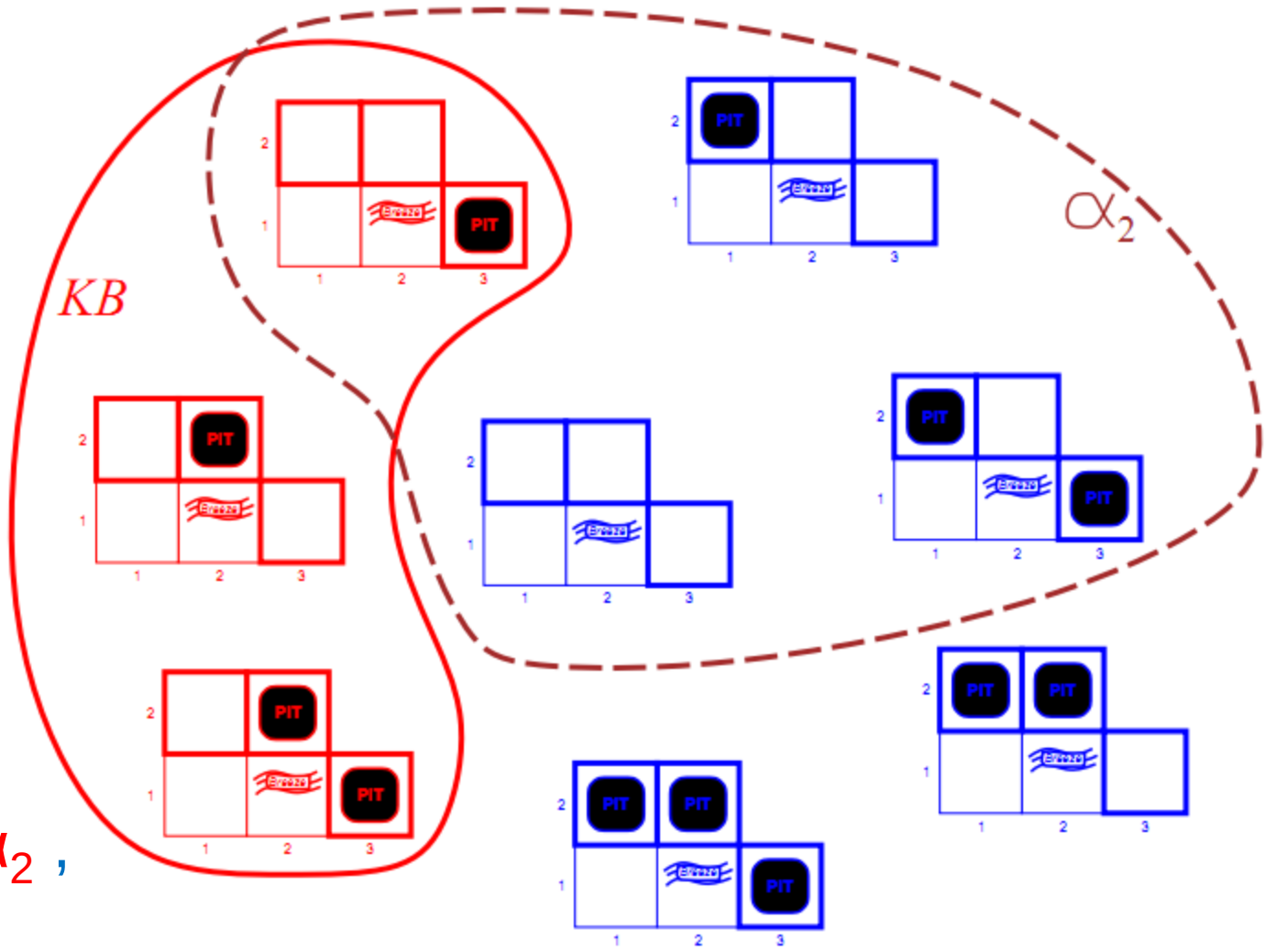
- KB = wumpus-world rules + observations



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

Wumpus models

- KB = wumpus-world rules + observations



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

Inference – Suy diễn

- An inference procedure can
  - Generate new sentences entailed by KB
  - Determine whether or not a given sentence is entailed by KB (e.g "prove")
- $KB \vdash_i \alpha$
  - sentence α can be derived from KB by inference procedure i
- Soundness (tính đúng đắn): i is sound if

  whenever $KB \vdash_i \alpha$, it is also true that $KB \models \alpha$
- Completeness (tính đầy đủ): i is complete if

  whenever $KB \models \alpha$, it is also true that $KB \vdash_i \alpha$

Inference – Suy diễn (cont.)

- Our Goal is to define a simple logic:
  - expressive enough to say almost anything of interest,
  - but also has a sound and complete inference procedure.
  - allow an agent to answer any question whose answer follows from what is known by the KB.

Propositional logic

Propositional logic: Syntax

- Propositional logic is the simplest logic - illustrates basic ideas
- The proposition symbols S, S_1, S_2, etc. are sentences
- If S, S_1, S_2 are sentences, then
  - $\neg$ S is a sentence (negation)
  - $S_1 \wedge S_2$ is a sentence (conjunction)
  - $S_1 \vee S_2$ is a sentence (disjunction)
  - $S_1 \rightarrow S_2$ is a sentence (implication)
  - $S_1 \leftrightarrow S_2$ is a sentence (biconditional)

Propositional logic – Syntax

- A BNF (Backus–Naur Form) grammar of sentences in propositional logic.

Sentence	=	AtomicSentence | ComplexSentence
AtomicSentence	=	True | False | Symbol
Symbol	=	P | Q | R | …
ComplexSentence	=	¬Sentence
		| (Sentence ∧ Sentence)
		| (Sentence ∨ Sentence)
		| (Sentence → Sentence)
		| (Sentence ↔ Sentence)

Precedence of logical operators

- Precedence of logical operators helps us to decide which operator will get evaluated first in a complicated looking compound proposition
- The precedence is: ¬, ∧, ∨, →, ↔
- One can use "()" to determine the precedence of logical operators
- Examples:
  - p ∧ q ∨ r is equivalent to (p ∧ q) ∨ r ; not p ∧ (q ∨ r)
  - ¬p ∧ q is equivalent to (¬p) ∧ q; not ¬(p ∧ q)
  - p ∧¬q → r is equivalent to (p ∧ (¬q)) → r; not p ∧ (¬(q → r)) or not p ∧ ((¬q) → r)

Propositional logic: Semantics

- Each model specifies true/false for each proposition symbol
- E.g. $P_{1;2}$, $P_{2;2}$, $P_{3;1}$ ➔ true, 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		

Example:

$P_{1;2} \wedge (P_{2;2} \vee P_{3;1})$ = true $\wedge$ (false $\vee$ true)=true $\wedge$ true=true

Propositional logic: Translation

- It is not sunny this afternoon and it is colder than yesterday.
- I will go swimming only if it is sunny.
- If I do not go swimming then I will play football.
- If I play football, then I will be home by sunset.

Propositional logic: Translation (cont.)

- Suppose, we have the following premises: Let
  - p be “It is sunny this afternoon”
  - q be “It is colder yesterday” r be “I will go swimming”
  - s be “I will play football” t be “I will be home by sunset”
- Representation:
  - “It is not sunny this afternoon and it is colder than yesterday”: $\neg p \wedge q$
  - “I will go swimming only if it is sunny”: $p \rightarrow r$
  - “If I do not go swimming then I will play football”: $\neg r \rightarrow s$
  - “If I play football, then I will be home by sunset”: $s \rightarrow t$

Truth tables for connectives

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \rightarrow Q$	$P \leftrightarrow Q$
false	false					
false	true					
true	false					
true	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

Logical equivalence

- Two sentences are logically equivalent iff true in same models ($\alpha \equiv \beta$, or $\alpha = \beta$, or $\alpha \leftrightarrow \beta$); $\alpha \equiv \beta$ iff $\alpha \models \beta$ and $\beta \models \alpha$.

- Commutativity $\alpha \wedge \beta \equiv \beta \wedge \alpha$
  $\alpha \vee \beta \equiv \beta \vee \alpha$
- Associativity $(\alpha \wedge \beta) \wedge \gamma \equiv \alpha \wedge (\beta \wedge \gamma)$
  $(\alpha \vee \beta) \vee \gamma \equiv \alpha \vee (\beta \vee \gamma)$
- Double-negation elimination $\neg\neg\alpha \equiv \alpha$
- Contraposition $\alpha \rightarrow \beta \equiv \neg\beta \rightarrow \neg\alpha$
- Implication elimination $\alpha \rightarrow \beta \equiv \neg\alpha \vee \beta$

Logical equivalence (cont.)

- Two sentences are logically equivalent iff true in same models ($\alpha \equiv \beta$, or $\alpha = \beta$, or $\alpha \leftrightarrow \beta$); $\alpha \equiv \beta$ iff $\alpha \models \beta$ and $\beta \models \alpha$.

- Bicondition elimination $\alpha \leftrightarrow \beta \equiv (\alpha \rightarrow \beta) \wedge (\beta \rightarrow \alpha)$
- De Morgan $\neg(\alpha \wedge \beta) \equiv \neg\alpha \vee \neg\beta$
  $\neg(\alpha \vee \beta) \equiv \neg\alpha \wedge \neg\beta$
- Distributivity $\alpha \wedge (\beta \vee \gamma) \equiv (\alpha \wedge \beta) \vee (\alpha \wedge \gamma)$
  $\alpha \vee (\beta \wedge \gamma) \equiv (\alpha \vee \beta) \wedge (\alpha \vee \gamma)$

Validity and satisfability – khả thỏa

- A sentence is valid (hằng đúng) 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 satisfable (khả thỏa) if it is true in some model

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

- A sentence is unsatisable if it is true in no models

e.g., $A \wedge \neg A$

- Satisability is connected to inference via the following:
  - $KB \models \alpha$ if and only if $(KB \wedge \neg\alpha)$ is unsatisfable
  - i.e., prove α by reductio ad absurdum

Example

Prove $((p \vee q) \wedge \neg p) \models q$

Case 1: Prove $((p \vee q) \wedge \neg p) \rightarrow q$ is valid

p	q	¬p	p ∨ q	(p ∨ q) ∧ ¬p	((p ∨ q) ∧ ¬p) → q
false	false	true	false	false	true
false	true	true	true	true	true
true	false	false	true	false	true
true	true	false	true	false	true

Case 2: $((p \vee q) \wedge \neg p) \wedge \neg q$ is unsatisable

p	q	¬p	p ∨ q	(p ∨ q) ∧ ¬p	((p ∨ q) ∧ ¬p) ∧ ¬q
false	false	true	false	false	false
false	true	true	true	true	false
true	false	false	true	false	false
true	true	false	true	false	false

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 alg.
    - Typically require translation of sentences into a normal form
  - Model checking
    - truth table enumeration (always exponential in n)
    - improved backtracking, e.g., Davis – Putnam – Logemann - Loveland
    - heuristic search in model space (sound but incomplete) e.g., min-conficts-like hill-climbing algorithms

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].

  R1: $\neg P_{1,1}$

  R2: $\neg B_{1,1}$

  R3: $B_{2,1}$

- "Pits cause breezes in adjacent squares"

  R4: $B_{1,1} \leftrightarrow (P_{1,2} \vee P_{2,1})$

  R5: $B_{2,1} \leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$

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

Truth tables for inference

$B_{1,1}$	$B_{2,1}$	$P_{1,1}$	$P_{1,2}$	$P_{2,1}$	$P_{2,2}$	$P_{3,1}$	R_1	R_2	R_3	R_4	R_5	KB
false	*false*	*false*	*false*	*false*	*false*	*false*	*true*	*true*	*true*	*true*	*false*	*false*
false	*false*	*false*	*false*	*false*	*false*	*true*	*true*	*true*	*false*	*true*	*false*	*false*
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
false	*true*	*false*	*false*	*false*	*false*	*false*	*true*	*true*	*false*	*true*	*true*	*false*
false	*true*	*false*	*false*	*false*	*false*	*true*	*true*	*true*	*true*	*true*	*true*	*true*
false	*true*	*false*	*false*	*false*	*true*	*false*	*true*	*true*	*true*	*true*	*true*	*true*
false	*true*	*false*	*false*	*false*	*true*	*true*	*true*	*true*	*true*	*true*	*true*	*true*
false	*true*	*false*	*false*	*true*	*false*	*false*	*true*	*false*	*false*	*true*	*true*	*false*
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
true	*true*	*true*	*true*	*true*	*true*	*true*	*false*	*true*	*true*	*false*	*true*	*false*

- KB is true if R_1, …, R_5 are true; only 3 rows satisfy (3 in 128 rows)
- $\alpha_1 = \neg P_{1,2}$ is true, no pit in [1,2]. Thus, KB $\models \alpha_1$.
- $\alpha_2 = \neg P_{2,2}$ is true in 1 row and false in 2 rows. Thus, the agent cannot conclude that there is no pit in [2,2], KB $\not\models \alpha_2$.

Inference by enumeration

- Depth-first enumeration of all models is sound and complete

PL-TRUE? returns true if a sentence holds within a model

```
function TT-ENTAILS?(KB, α) returns true or false
    inputs: KB, the knowledge base, a sentence in propositional logic
            α, the query, a sentence in propositional logic

    symbols ← a list of the proposition symbols in KB and α
    return TT-CHECK-ALL(KB, α, symbols, [])

---

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  // when KB is false, always return true
    else do
        P ← FIRST(symbols); rest ← REST(symbols)
        return TT-CHECK-ALL(KB, α, rest, EXTEND(P, true, model)) and
               TT-CHECK-ALL(KB, α, rest, EXTEND(P, false, model))
```

Inference by enumeration

- PL-TRUE? returns true if a sentence holds within a model
- **`EXTEND(P, true, model)`** ≡ model ∪ {P = true}
- The keyword “and” is used here as a logical operation on its two arguments
- $O(2^n)$ for n symbols; problem is co-NP-complete

Propositional theorem proving

Rules of inference for propositional logic

1. And-Elimination (Luật rút gọn)

$$\frac{\alpha_1 \wedge \alpha_2 \wedge \ldots \wedge \alpha_n}{\therefore \alpha_1}$$

2. Or-Introduction (Luật cộng)

$$\frac{\alpha_1}{\therefore \alpha_1 \vee \alpha_2 \vee \ldots \vee \alpha_n}$$

3. And-Introduction (Luật đưa vào liên kết VÀ)

$$\frac{\alpha_1, \alpha_2, \ldots, \alpha_n}{\therefore \alpha_1 \wedge \alpha_2 \wedge \ldots \wedge \alpha_n}$$

4. Modus Ponens (Luật khẳng định)

$$\frac{\alpha \rightarrow \beta, \quad \alpha}{\therefore \beta}$$

5. Modus Tollens (Luật phủ định)

$$\frac{\alpha \rightarrow \beta, \quad \neg\beta}{\therefore \neg\alpha}$$

6. Syllogism (Tam đoạn luận)

$$\frac{\alpha \rightarrow \beta, \quad \beta \rightarrow \gamma}{\therefore \alpha \rightarrow \gamma}$$

7. Disjunctive syllogism (Tam đoạn luận tuyển)

$$\frac{\alpha \vee \beta, \quad \neg\alpha}{\therefore \beta}$$

8. Resolution (Hợp giải)

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

Wumpus world

- KB including:

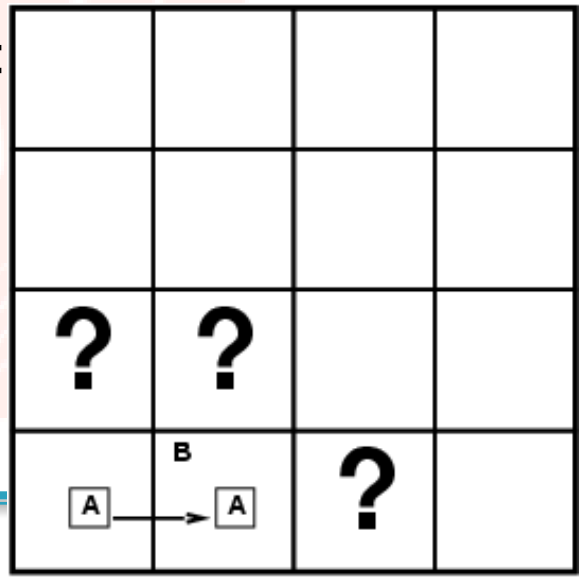
B: Breeze
P: Pit

$R1 : \neg P_{1,1}$ $R2 : \neg B_{1,1}$ $R3 : B_{2,1}$

$R4 : B_{1,1} \leftrightarrow (P_{1,2} \vee P_{2,1})$ $R5 : B_{2,1} \leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$

Prove: $\neg \mathbf{P_{1,2}}$ = "There is no pit in [1,2]".

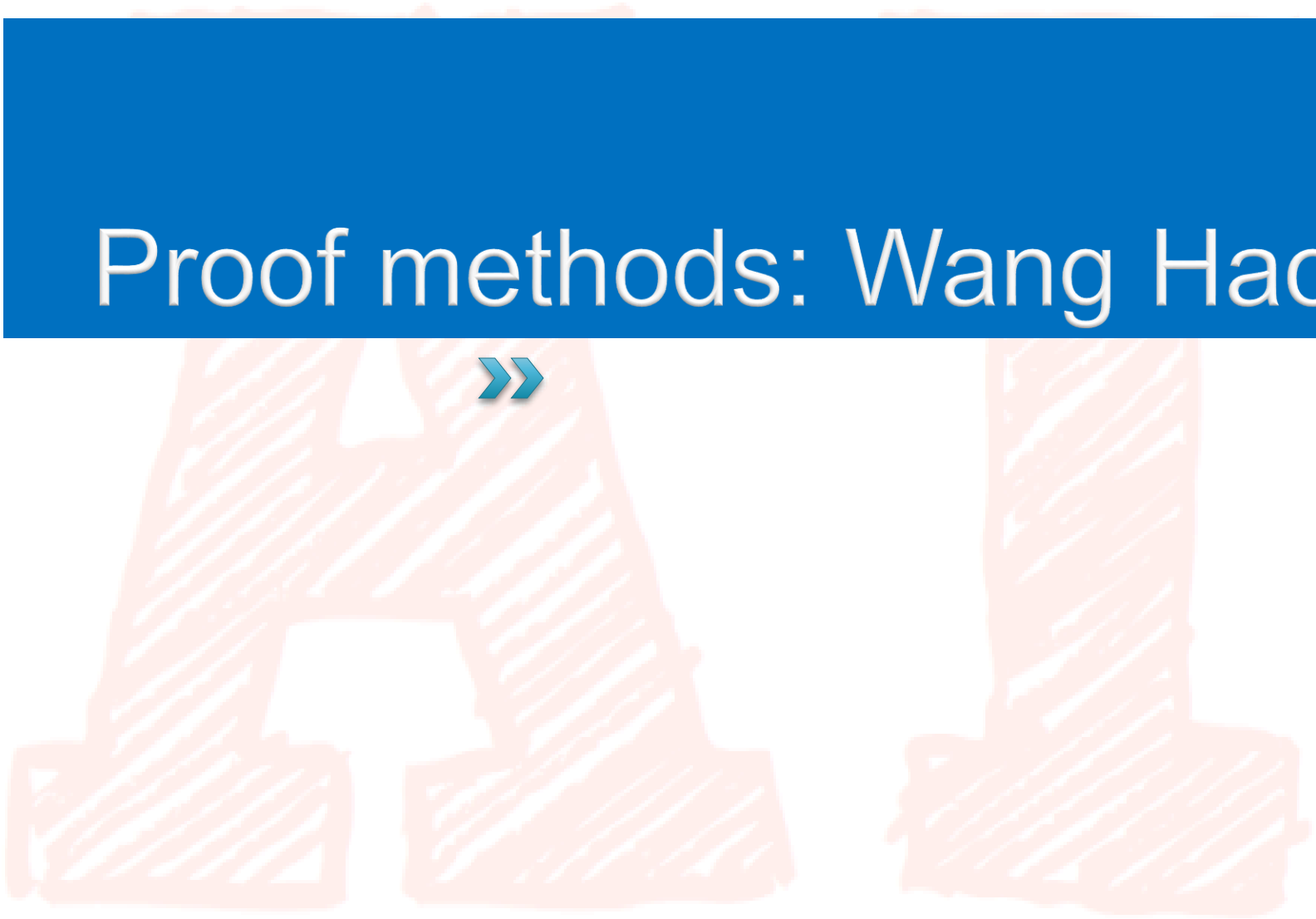
1. R4 is equivalent to
   $R6 : (B_{1,1} \rightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \rightarrow B_{1,1})$
2. Apply And-Elimination to R6: $R7 : ((P_{1,2} \vee P_{2,1}) \rightarrow B_{1,1})$
3. Logical equivalence for contrapositives gives R7:
   $R8 : (\neg B_{1,1} \rightarrow \neg(P_{1,2} \vee P_{2,1}))$
4. Apply Modus Ponens to R8 and R2,
   $R9 : \neg(P_{1,2} \vee P_{2,1})$
5. Apply de Morgan, : $R10 : \neg P_{1,2} \wedge \neg P_{2,1}$
   In conclusion: $\neg P_{1,2}$



Exercise

- KB: {p ∧ q, p → r, (q ∧ r) → s}. **Prove: s**
- KB: {p ∧ q, p → ¬(q ∧ r),s → r}. **Prove: ¬s**
- KB: {¬ (p ∨ q) → r, ¬p, ¬r}. **Prove: q**
- KB: {¬ p → (q ∧ r), p →s, ¬s}. **Prove: q**

Proof methods: Wang Hao



Proof methods: Wang Hao Algorithm

1. Write the premises down separated by commas, followed by an arrow and then the theorem to be proved :

   $p_1, p_2, ..., p_n \rightarrow q_1, q_2, ..., q_m$

   - where p_i and q_i : propositions formed by using connectives ($\neg, \wedge, \vee$)

2. If one of the formulae separated by commas is the negation of a formula, drop the negation sign and move it to the other side of the arrow.

   - Ex.: $(p \vee q, \neg(r \vee s), \neg g, p \wedge \neg r) \rightarrow (s, \neg q)$
     $\equiv (p \vee q, p \wedge \neg r, q) \rightarrow (r \vee s, g, s)$

3. If the principal connective of a formula on the left is $\wedge$ (and), or on the right of the arrow is v (or), replace the connective by a comma.

   - Ex.: $(p \vee q, p \wedge \neg r, q) \rightarrow (r \vee s, g, s)$
     $\equiv (p \vee q, p, \neg r, q) \rightarrow (r, s, g, s)$

Proof methods: Wang Hao Algorithm

4. If the principal connective of a formula on the left is *v* (or), or on the right of the arrow is $\wedge$ (and), then produce two new lines
   - Ex. : $(p \vee q, p, \neg r, q) \rightarrow (r, s, g, s)$ is changed to the two lines $(p, p, \neg r, q) \rightarrow (r, s, g, s)$ and $(q, p, \neg r, q) \rightarrow (r, s, g, s)$
5. If the same formula ever appears on both sides of an arrow, the line is proved
   - Ex. : $p, q \rightarrow q$ is proved
     $p, \neg p \rightarrow q \equiv p \rightarrow p, q$ is proved
6. If no connectives (~, $\wedge$ or v) remain in a line, and no proposition appears on both sides of the arrow, the line cannot be proved.
7. A problem is proved if all lines derived from the original form are proved.

Example

- KB = {$p \rightarrow q$, $\neg q$, $\neg r$}. Prove: $\neg(p \vee r)$
- KB = {$p \wedge (\neg p \vee q)$}. Prove: q
- KB = {$(p \vee q) \wedge (\neg p \vee r$}. Prove: $(q \vee r)$

Example (cont.)

KB = $\{p \rightarrow q, \neg q, \neg r\}$. Prove: $\neg(p \vee r)$

$(p \rightarrow q, \neg q, \neg r) \rightarrow (\neg(p \vee r))$

$\equiv (\neg p \vee q, \neg q, \neg r) \rightarrow (\neg(p \vee r))$

$\equiv (\neg p \vee q, p \vee r) \rightarrow (q, r)$

$\equiv (\neg p, p \vee r) \rightarrow (q, r)$
$(q, p \vee r) \rightarrow (q, r)$

$\equiv (p) \rightarrow (p, q, r)$
$(r) \rightarrow (p, q, r)$
$(q, p) \rightarrow (q, r)$
$(q, r) \rightarrow (q, r)$

Proof Methods: Resolution

Proof Methods: Resolution

- Conjunctive Normal Form (CNF|universal)
  - conjunction of disjunctions of literals

Clauses

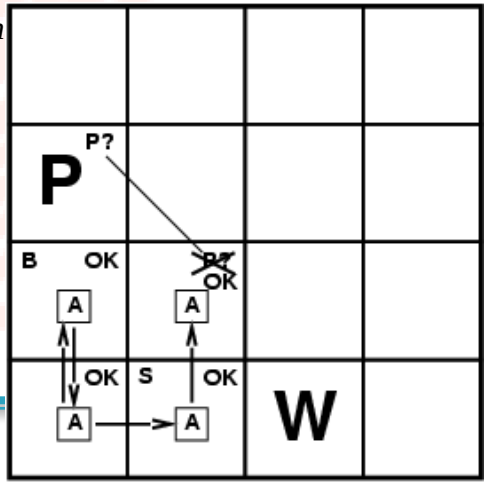
  - E.g., $(A \vee \neg B) \wedge (B \vee \neg C \vee \neg D)$
- Resolution inference rule (for CNF): complete for propositional logic

$$\frac{\alpha_1 \vee \ldots \vee \alpha_n, \quad \beta_1 \vee \ldots \vee \beta_m}{\therefore \alpha_1 \vee \ldots \vee \alpha_{i-1} \vee \alpha_{i+1} \vee \ldots \vee \alpha_n \vee \beta_1 \vee \ldots \vee \beta_{j-1} \vee \beta_{j+1} \vee \ldots \vee \beta_m}$$

Where $\alpha_i = \neg\beta_j$, e.g.,

$$\frac{P_{1,3} \vee P_{2,2}\ , \ \neg P_{2,2}}{\therefore P_{1,3}}$$

Resolution is sound and complete for propositional logic



Proof Methods: Resolution

- Resolution rule (for short):

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

- Resolution refutation:
  - Convert all sentences to CNF
  - Negate the desired conclusion (converted to CNF)
  - Apply resolution rule (to each pair of clauses) until either
    - Derive false (a contradiction) – existing {P, ¬ P} in KB
    - Can't apply any more

Notice that: If the principal connective of a formula on the left is ∧ (and), or on the right of the arrow is v (or), replace the connective by a comma.

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 :

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

Exercise

- Conversion to CNF for the following sentences:

1. $\neg(P \rightarrow Q)$
2. $\neg(P \vee \neg Q) \wedge (P \vee Q)$
3. $(\neg P \wedge Q) \rightarrow R$
4. $\neg(P \wedge Q) \wedge (P \vee Q)$
5. $(P \rightarrow Q) \rightarrow R$
6. $P \rightarrow ((Q \wedge R) \rightarrow S)$
7. $P \vee (\neg P \wedge Q \wedge R)$
8. $\neg(P \rightarrow Q) \vee (P \vee Q)$
9. $(\neg P \wedge Q) \vee (P \wedge \neg Q)$

Resolution algorithm

- Proof by contradiction, i.e., show $KB \wedge \neg\alpha$: unsatisfiable

function PL-RESOLUTION(KB, α) **returns** *true* or *false*

inputs: KB, the knowledge base, a sentence in propositional logic
α, the query, a sentence in propositional logic

clauses ← the set of clauses in the CNF representation of $KB \wedge \neg\alpha$
new ← { }
loop do
  for each C_i, C_j **in** *clauses* **do**
    resolvents ← PL-RESOLVE(C_i, C_j)
    if *resolvents* contains the empty clause **then return** *true*
    new ← *new* ∪ *resolvents*
  if *new* ⊆ *clauses* **then return** *false*
  clauses ← *clauses* ∪ *new*

Resolution example

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

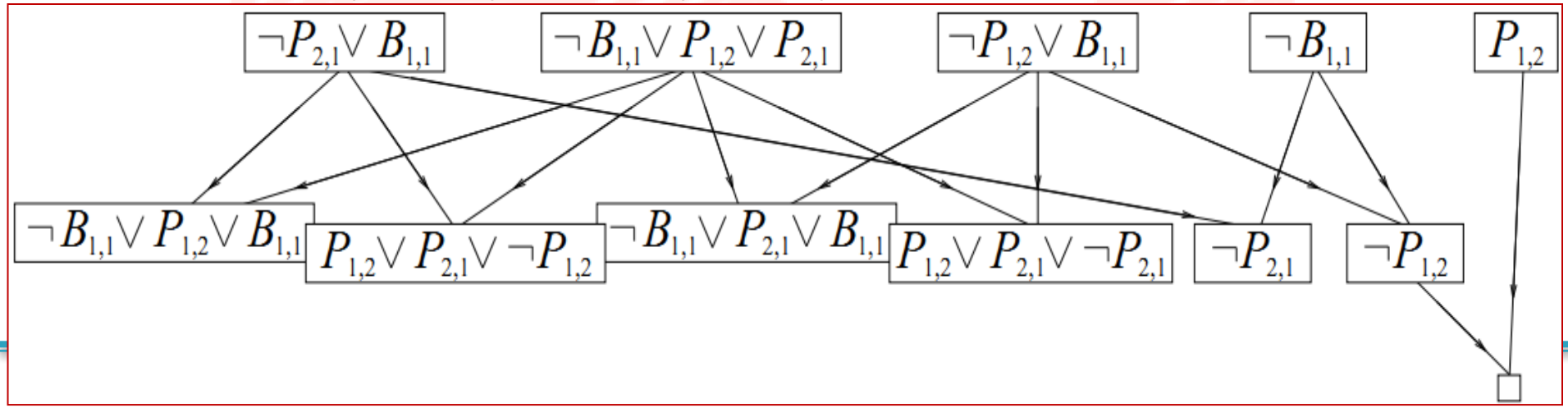
Conversion to CNF:

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

Prove: $\alpha = \neg P_{1,2}$

Then:

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



Example

- KB: {$p \wedge q$, $p \rightarrow r$, $(q \wedge r) \rightarrow s$}. Prove: s
- Conversion to CNF:
  - KB: {$p \wedge q$, $\neg p \vee r$, $\neg (q \wedge r) \vee s$}
  - KB: {p, q, $\neg p \vee r$, $\neg q \vee \neg r \vee s$}
- Add $\neg$ s to KB
  - KB: {$p \wedge q$, $\neg p \vee r$, $\neg q \vee \neg r \vee s$, $\neg s$}

Literal	Sentences	Note
	p , q, $\neg p \vee r$, $\neg q \vee \neg r \vee s$, $\neg s$	

Example

- KB: {$p \wedge q$, $p \rightarrow r$, $(q \wedge r) \rightarrow s$}. Prove: s
- Conversion to CNF:
  - KB: {$p \wedge q$, $\neg p \vee r$, $\neg (q \wedge r) \vee s$}
  - KB: {p, q, $\neg p \vee r$, $\neg q \vee \neg r \vee s$}
- Add $\neg$ s to KB
  - KB: {$p \wedge q$, $\neg p \vee r$, $\neg q \vee \neg r \vee s$, $\neg s$}

Literal	Sentences	Note
	p , q, $\neg p \vee r$, $\neg q \vee \neg r \vee s$, $\neg s$	
p		

Example

- KB: {$p \wedge q$, $p \rightarrow r$, $(q \wedge r) \rightarrow s$}. Prove: s
- Conversion to CNF:
  - KB: {$p \wedge q$, $\neg p \vee r$, $\neg (q \wedge r) \vee s$}
  - KB: {p, q, $\neg p \vee r$, $\neg q \vee \neg r \vee s$}
- Add $\neg$ s to KB
  - KB: {$p \wedge q$, $\neg p \vee r$, $\neg q \vee \neg r \vee s$, $\neg s$}

Literal	Sentences	Note
	p , q, $\neg p \vee r$, $\neg q \vee \neg r \vee s$, $\neg s$	
p	p , q, $\neg p \vee r$, $\neg q \vee \neg r \vee s$, $\neg s$	

Example

- KB: {$p \wedge q$, $p \rightarrow r$, $(q \wedge r) \rightarrow s$}. Prove: s
- Conversion to CNF:
  - KB: {$p \wedge q$, $\neg p \vee r$, $\neg (q \wedge r) \vee s$}
  - KB: {p, q, $\neg p \vee r$, $\neg q \vee \neg r \vee s$}
- Add $\neg$ s to KB
  - KB: {$p \wedge q$, $\neg p \vee r$, $\neg q \vee \neg r \vee s$, $\neg s$}

Literal	Sentences	Note
	p , q, $\neg p \vee r$, $\neg q \vee \neg r \vee s$, $\neg s$	
p	p , q, $\neg p \vee r$, $\neg q \vee \neg r \vee s$, $\neg s$	
	q, r, $\neg q \vee \neg r \vee s$, $\neg s$	

Example

- KB: {$p \wedge q$, $p \rightarrow r$, $(q \wedge r) \rightarrow s$}. Prove: s
- Conversion to CNF:
  - KB: {$p \wedge q$, $\neg p \vee r$, $\neg (q \wedge r) \vee s$}
  - KB: {p, q, $\neg p \vee r$, $\neg q \vee \neg r \vee s$}
- Add $\neg$ s to KB
  - KB: {$p \wedge q$, $\neg p \vee r$, $\neg q \vee \neg r \vee s$, $\neg s$}

Literal	Sentences	Note
	p , q, $\neg p \vee r$, $\neg q \vee \neg r \vee s$, $\neg s$	
p	p , q, $\neg p \vee r$, $\neg q \vee \neg r \vee s$, $\neg s$	
q	q, r, $\neg q \vee \neg r \vee s$, $\neg s$	

Example

- KB: {$p \wedge q$, $p \rightarrow r$, $(q \wedge r) \rightarrow s$}. Prove: s
- Conversion to CNF:
  - KB: {$p \wedge q$, $\neg p \vee r$, $\neg (q \wedge r) \vee s$}
  - KB: {p, q, $\neg p \vee r$, $\neg q \vee \neg r \vee s$}
- Add $\neg$ s to KB
  - KB: {$p \wedge q$, $\neg p \vee r$, $\neg q \vee \neg r \vee s$, $\neg s$}

Literal	Sentences	Note
	p , q, $\neg p \vee r$, $\neg q \vee \neg r \vee s$, $\neg s$	
p	p , q, $\neg p \vee r$, $\neg q \vee \neg r \vee s$, $\neg s$	
q	q, r, $\neg q \vee \neg r \vee s$, $\neg s$	
	r, $\neg r \vee s$, $\neg s$	

Example

- KB: {$p \wedge q$, $p \rightarrow r$, $(q \wedge r) \rightarrow s$}. Prove: s
- Conversion to CNF:
  - KB: {$p \wedge q$, $\neg p \vee r$, $\neg (q \wedge r) \vee s$}
  - KB: {p, q, $\neg p \vee r$, $\neg q \ \vee \neg r \vee s$}
- Add $\neg$ s to KB
  - KB: {$p \wedge q$, $\neg p \vee r$, $\neg q \ \vee \neg r \vee s$, $\neg s$}

Literal	Sentences	Note
	p , q, $\neg p \vee r$, $\neg q \ \vee \neg r \vee s$, $\neg s$	
p	p , q, $\neg p \vee r$, $\neg q \ \vee \neg r \vee s$, $\neg s$	
q	q, r, $\neg q \ \vee \neg r \vee s$, $\neg s$	
r	r, $\neg r \vee s$, $\neg s$	

Example

- KB: {$p \wedge q$, $p \rightarrow r$, $(q \wedge r) \rightarrow s$}. Prove: s
- Conversion to CNF:
  - KB: {$p \wedge q$, $\neg p \vee r$, $\neg (q \wedge r) \vee s$}
  - KB: {p, q, $\neg p \vee r$, $\neg q \vee \neg r \vee s$}
- Add $\neg$ s to KB
  - KB: {$p \wedge q$, $\neg p \vee r$, $\neg q \vee \neg r \vee s$, $\neg s$}

Literal	Sentences	Note
	p , q, $\neg p \vee r$, $\neg q \vee \neg r \vee s$, $\neg s$	
p	p , q, $\neg p \vee r$, $\neg q \vee \neg r \vee s$, $\neg s$	
q	q, r, $\neg q \vee \neg r \vee s$, $\neg s$	
r	r, $\neg r \vee s$, $\neg s$	
	s, $\neg s$	

Example

- KB: {$p \wedge q$, $p \rightarrow r$, $(q \wedge r) \rightarrow s$}. Prove: s
- Conversion to CNF:
  - KB: {$p \wedge q$, $\neg p \vee r$, $\neg (q \wedge r) \vee s$}
  - KB: {p, q, $\neg p \vee r$, $\neg q \ \vee \neg r \vee s$}
- Add $\neg$ s to KB
  - KB: {$p \wedge q$, $\neg p \vee r$, $\neg q \ \vee \neg r \vee s$, $\neg s$}

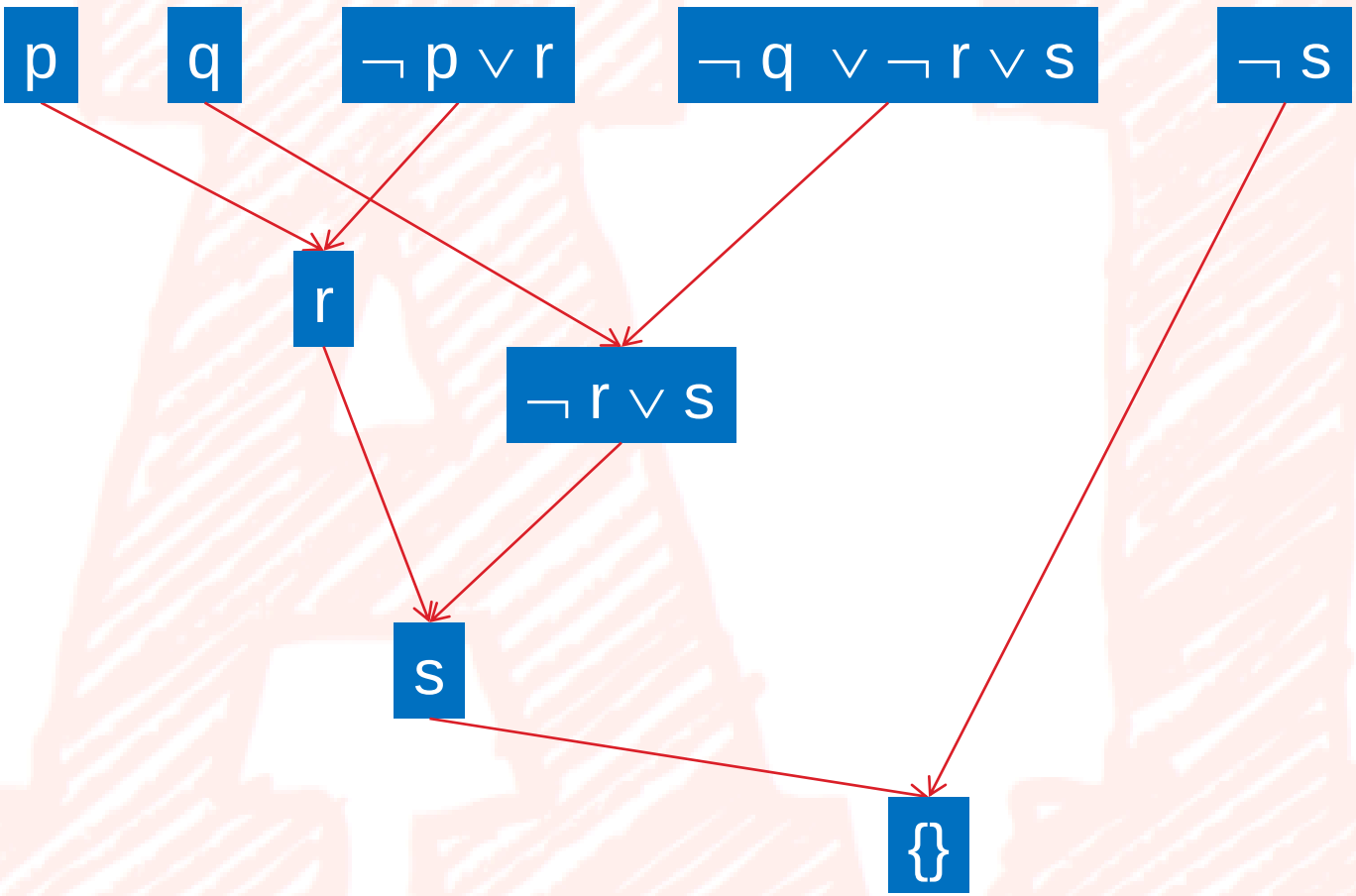
Literal	Sentences	Note
	p , q, $\neg p \vee r$, $\neg q \ \vee \neg r \vee s$, $\neg s$	
p	p , q, $\neg p \vee r$, $\neg q \ \vee \neg r \vee s$, $\neg s$	
q	q, r, $\neg q \ \vee \neg r \vee s$, $\neg s$	
r	r, $\neg r \vee s$, $\neg s$	
s	s, $\neg s$	

Example

- KB: {$p \wedge q$, $p \rightarrow r$, $(q \wedge r) \rightarrow s$}. Prove: s
- Conversion to CNF:
  - KB: {$p \wedge q$, $\neg p \vee r$, $\neg (q \wedge r) \vee s$}
  - KB: {p, q, $\neg p \vee r$, $\neg q \ \vee \neg r \vee s$}
- Add $\neg$ s to KB
  - KB: {$p \wedge q$, $\neg p \vee r$, $\neg q \ \vee \neg r \vee s$, $\neg s$}

Literal	Sentences	Note
	p , q, $\neg p \vee r$, $\neg q \ \vee \neg r \vee s$, $\neg s$	
p	p , q, $\neg p \vee r$, $\neg q \ \vee \neg r \vee s$, $\neg s$	
q	q, r, $\neg q \ \vee \neg r \vee s$, $\neg s$	
r	r, $\neg r \vee s$, $\neg s$	
s	s, $\neg s$	
	{}	

Example (cont.)



Exercise 1

- KB: {¬p ∨ q, ¬q ∨ r, ¬r ∨ s, ¬u ∨ ¬s}
- Prove: ¬p ∨ ¬u

- KB = { A → B ∧ C, C → E ∨ F, B → ¬E ,A}
- Prove: F

- KB = {A → B; A → C ∨ E; B ∧ C → D; E → F; F ∨ D → G; A}
- Pove: E
- Prove: G

Forward and backward chaining

Horn Form KB

- Horn clause: a disjunction of literals of which exactly one is positive.
  - E.g.,: $(\neg P_{1,2} \vee \neg P_{2,1} \vee B_{1,1})$ is a Horn clause
    $(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1})$ is not a Horn clause
- Each Horn clause can be written as an implication :
  - $(\neg P_{1,2} \vee \neg P_{2,1} \vee B_{1,1})$ can be written as $(P_{1,2} \wedge P_{2,1}) \rightarrow B_{1,1}$
  - $(\neg W_{1,1} \vee \neg W_{1,2})$ can be written as $(W_{1,1} \wedge W_{1,2}) \rightarrow$ False
- Horn clause =
  - proposition symbol; or
  - (conjunction of symbols $\rightarrow$ symbol) (e.g., $P_1 \wedge P_2 \ldots \wedge P_n) \rightarrow Q$
- Horn KB = conjunction of Horn clauses

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

Forward vs. backward chaining

- FC is data-driven, cf. automatic, unconscious processing,

  e.g., object recognition, routine decisions

- May do lots of work that is irrelevant to the goal

- BC is goal-driven, appropriate for problem-solving,

  e.g., Where are my keys? How do I get into a PhD program?

- Complexity of BC can be much less than linear in size of KB

Forward and backward chaining

- Modus Ponens: complete for Horn KBs

$$\frac{\alpha_1, \alpha_2, \ldots, \alpha_n, \ \alpha_1 \wedge \alpha_2 \wedge \ldots \wedge \alpha_n \rightarrow \beta}{\therefore \beta}$$

- Can be used with forward chaining or backward chaining.
- These algorithms are very natural and run in linear time

Forward chaining

- For a given knowledge base, determine if a single proposition symbol Q is entailed.
- Idea:
  - fire any rule whose premises are satisfied in the KB,
  - add its conclusion to the KB, until query is found
- Assume the KB with the following rules and facts:

KB: R1: $A \wedge B \Rightarrow C$

R2: $C \wedge D \Rightarrow E$

R3: $C \wedge F \Rightarrow G$

F1: A

F2: B

F3: D

Prove: E

Forward chaining

- Prove: E

KB: R1: $A \wedge B \Rightarrow C$

R2: $C \wedge D \Rightarrow E$

R3: $C \wedge F \Rightarrow G$

F1: A

F2: B

F3: D

Rule R1 is satisfied.

F4: C

Forward chaining

- Prove: E

KB: R1: $A \wedge B \Rightarrow C$

R2: $C \wedge D \Rightarrow E$

R3: $C \wedge F \Rightarrow G$

F1: A

F2: B

F3: D

Rule R1 is satisfied.

F4: C

Rule R2 is satisfied.

F5: E

Forward chaining example

$P \rightarrow Q$ (1)

$L \wedge M \rightarrow P$ (2)

$B \wedge L \rightarrow M$ (3)

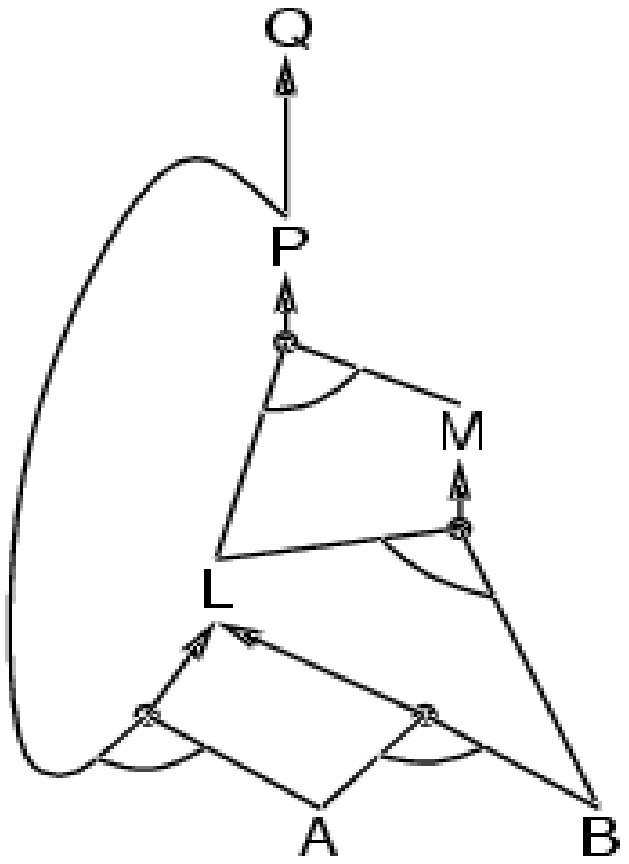
$A \wedge P \rightarrow L$ (4)

$A \wedge B \rightarrow L$ (5)

A (6)

B (7)

Prove: Q



Forward chaining example

Count the number of facts in the antecedent of the rule

$P \rightarrow Q$ (1)

$L \wedge M \rightarrow P$ (2)

$B \wedge L \rightarrow M$ (3)

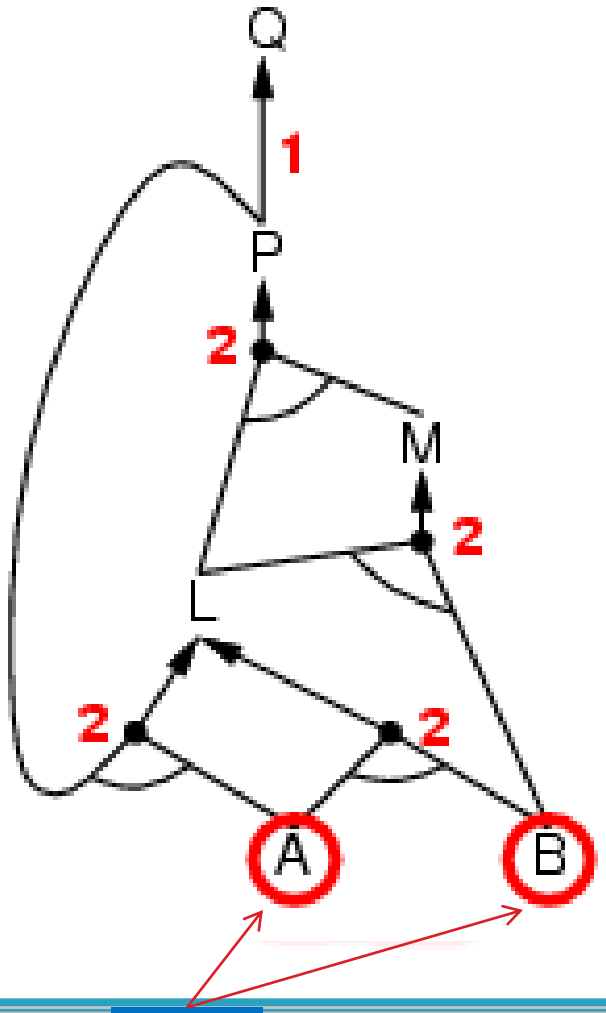
$A \wedge P \rightarrow L$ (4)

$A \wedge B \rightarrow L$ (5)

A (6)

B (7)

Prove: Q



Forward chaining example

Inferred facts decrease the count

$P \rightarrow Q$ (1)

$L \wedge M \rightarrow P$ (2)

$B \wedge L \rightarrow M$ (3)

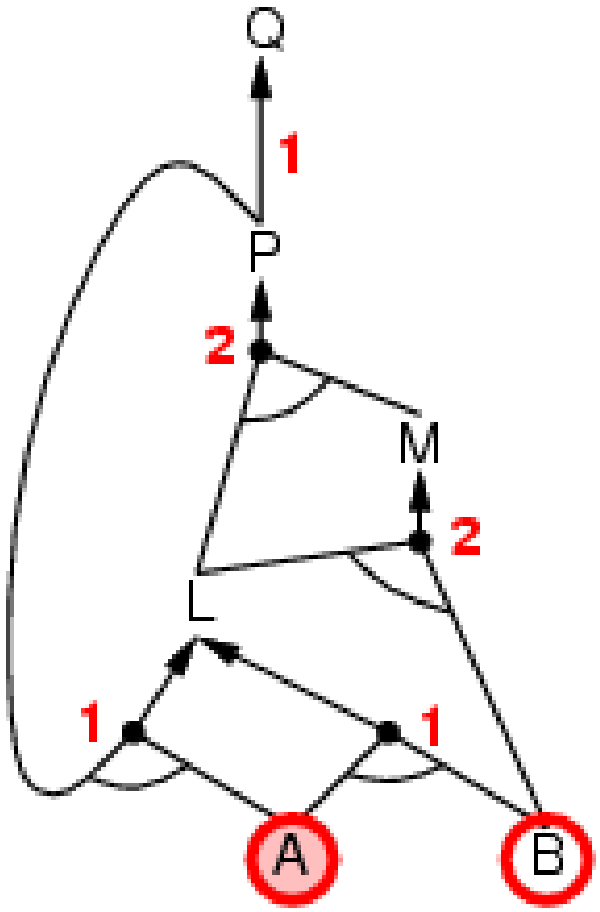
$A \wedge P \rightarrow L$ (4)

$A \wedge B \rightarrow L$ (5)

A (6)

B (7)

Prove: Q



Forward chaining example

New facts can be inferred when the count associated with a rule becomes 0

$P \rightarrow Q$ (1)

$L \wedge M \rightarrow P$ (2)

$B \wedge L \rightarrow M$ (3)

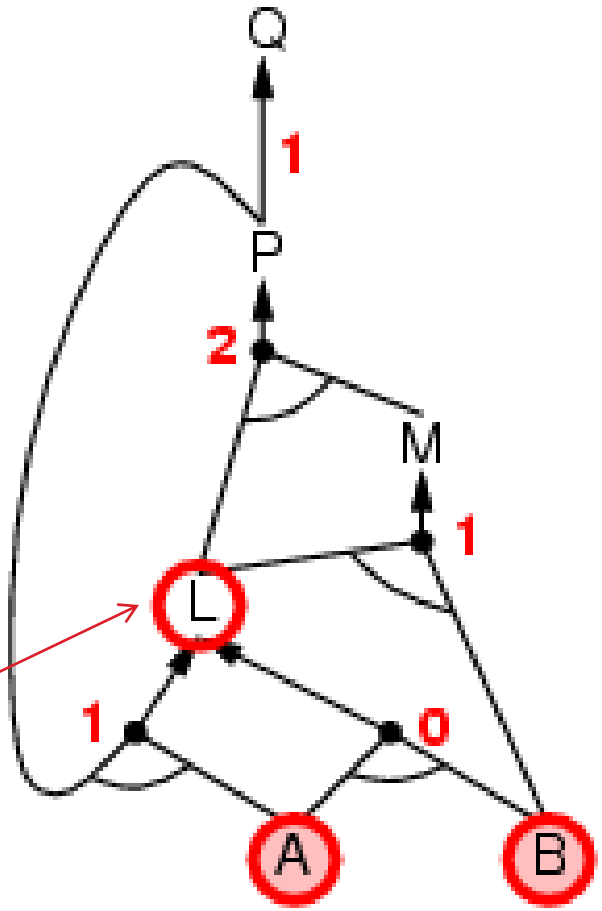
$A \wedge P \rightarrow L$ (4)

$A \wedge B \rightarrow L$ (5)

A (6)

B (7)

Prove: Q



Forward chaining example

$P \rightarrow Q$ (1)

$L \wedge M \rightarrow P$ (2)

$B \wedge L \rightarrow M$ (3)

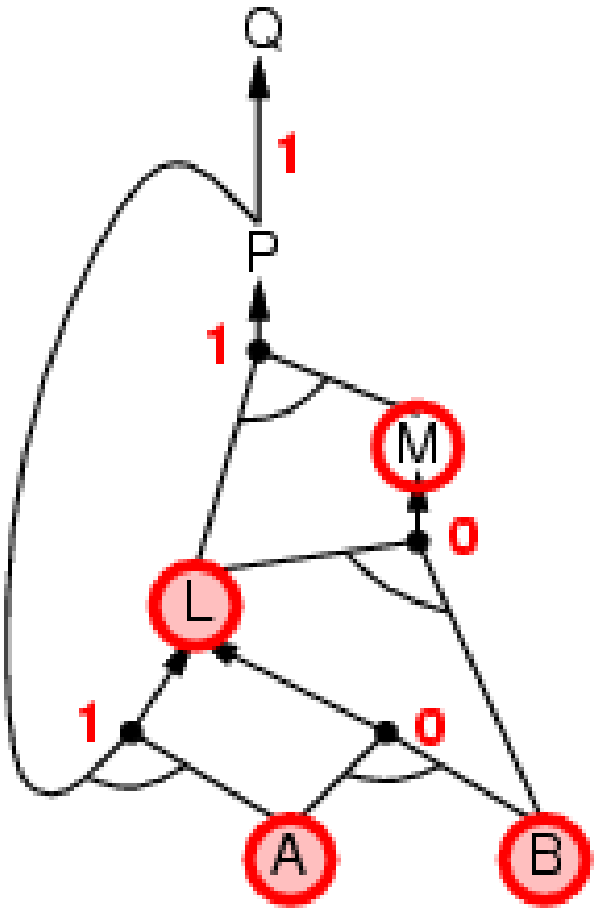
$A \wedge P \rightarrow L$ (4)

$A \wedge B \rightarrow L$ (5)

A (6)

B (7)

Prove: Q



Forward chaining example

$P \rightarrow Q$ (1)

$L \wedge M \rightarrow P$ (2)

$B \wedge L \rightarrow M$ (3)

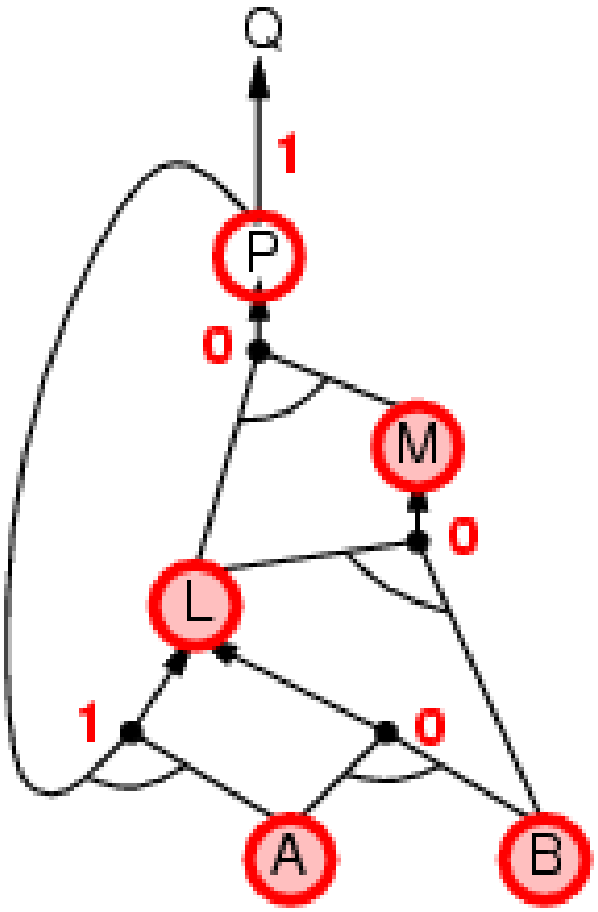
$A \wedge P \rightarrow L$ (4)

$A \wedge B \rightarrow L$ (5)

A (6)

B (7)

Prove: Q



Forward chaining example

$P \rightarrow Q$ (1)

$L \wedge M \rightarrow P$ (2)

$B \wedge L \rightarrow M$ (3)

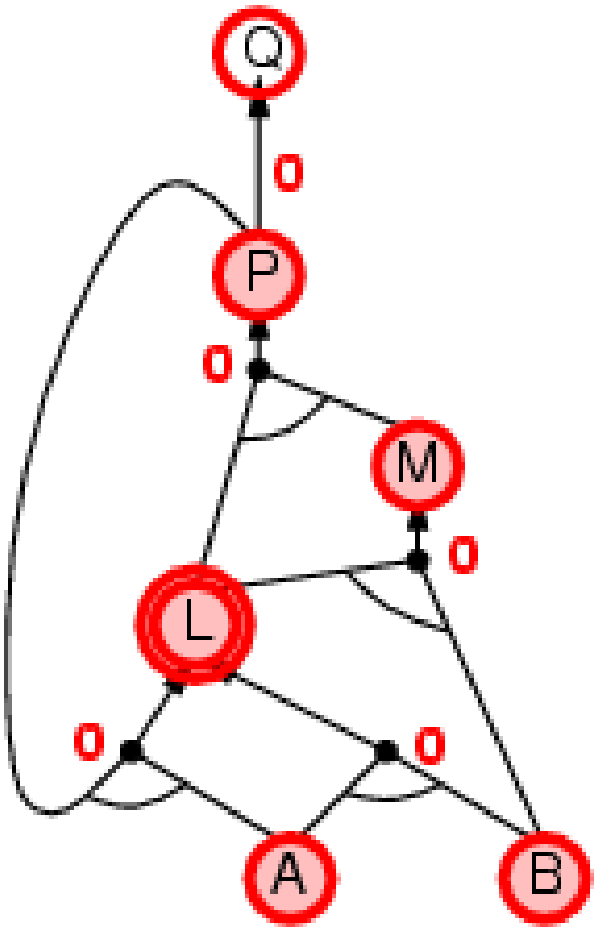
$A \wedge P \rightarrow L$ (4)

$A \wedge B \rightarrow L$ (5)

A (6)

B (7)

Prove: Q



Forward chaining example

$P \rightarrow Q$ (1)

$L \wedge M \rightarrow P$ (2)

$B \wedge L \rightarrow M$ (3)

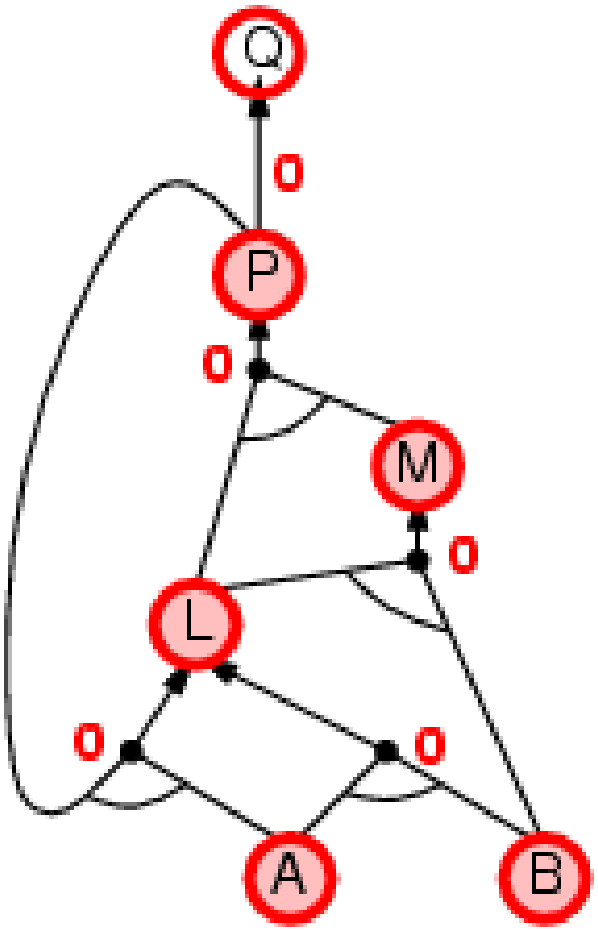
$A \wedge P \rightarrow L$ (4)

$A \wedge B \rightarrow L$ (5)

A (6)

B (7)

Prove: Q



Forward chaining example

$P \rightarrow Q$ (1)

$L \wedge M \rightarrow P$ (2)

$B \wedge L \rightarrow M$ (3)

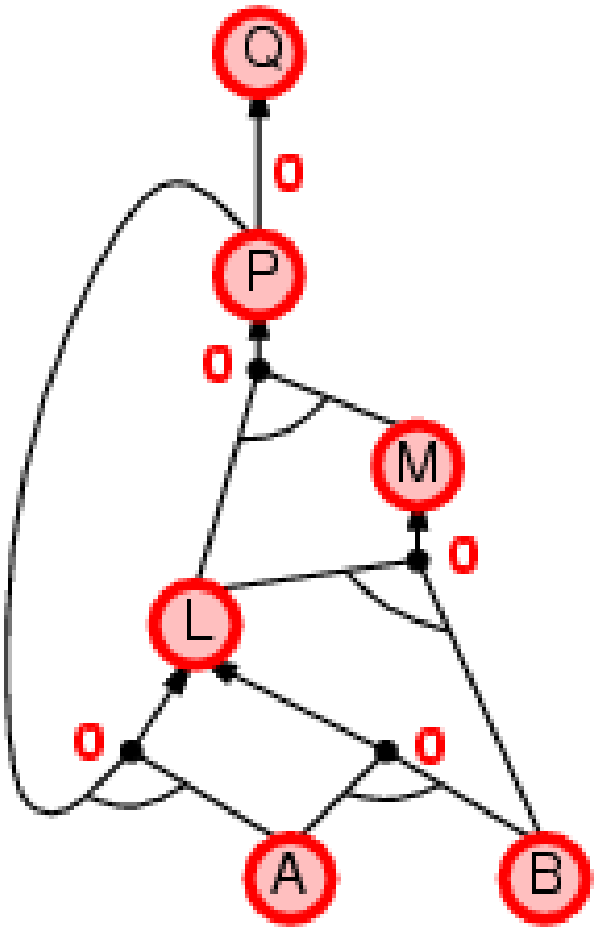
$A \wedge P \rightarrow L$ (4)

$A \wedge B \rightarrow L$ (5)

A (6)

B (7)

Prove: Q



Backward chaining (BC)

- Idea: work backwards from the query q:

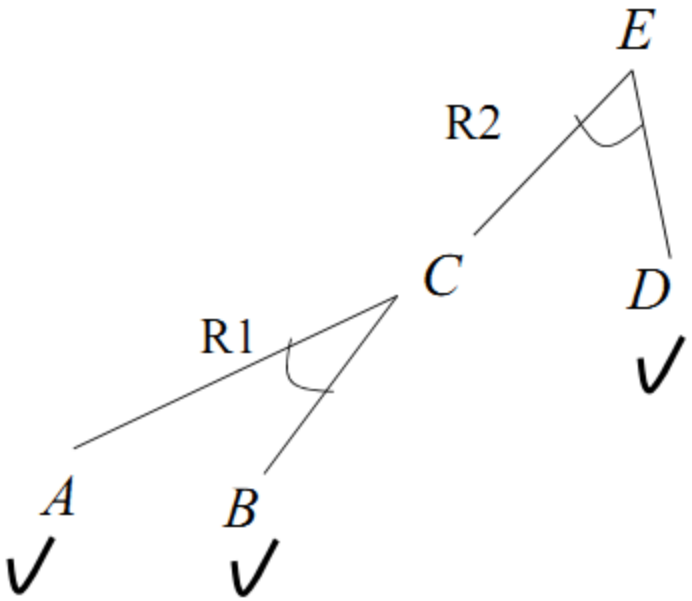
  to prove q by BC,

  - check if q is known already, or
  - prove by BC all premises of some rules concluding q

- Avoid loops: check if new subgoal is already on the goal stack
- Avoid repeated work: check if new subgoal
  - has already been proved true, or
  - has already failed

Backward chaining example

Prove: E



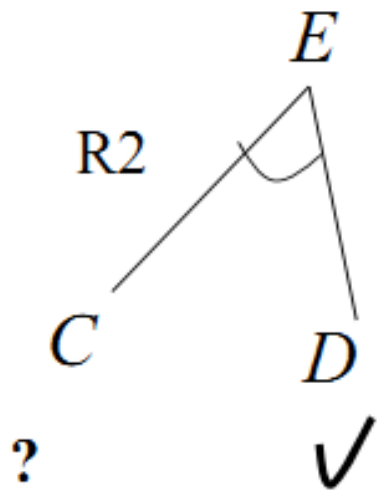
KB:
R1: $A \wedge B \Rightarrow C$
R2: $C \wedge D \Rightarrow E$
R3: $C \wedge F \Rightarrow G$

F1: A
F2: B
F3: D

- Backward chaining is more focused:
  - tries to prove the theorem only

Backward chaining example

Prove: E



KB:

R1: $A \wedge B \Rightarrow C$

R2: $C \wedge D \Rightarrow E$

R3: $C \wedge F \Rightarrow G$

F1: A

F2: B

F3: D

- Backward chaining is more focused:
  - tries to prove the theorem only

Backward chaining example

Efficient implementation

$P \Rightarrow Q$ ←

$L \wedge M \Rightarrow P$

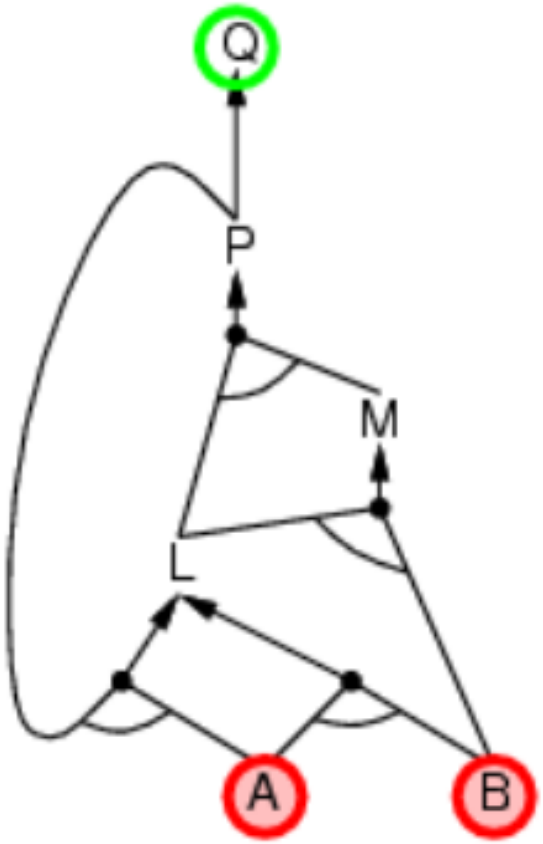
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A ←

B ←



Backward chaining example

Efficient implementation

$P \Rightarrow Q$ ←

$L \wedge M \Rightarrow P$ ←

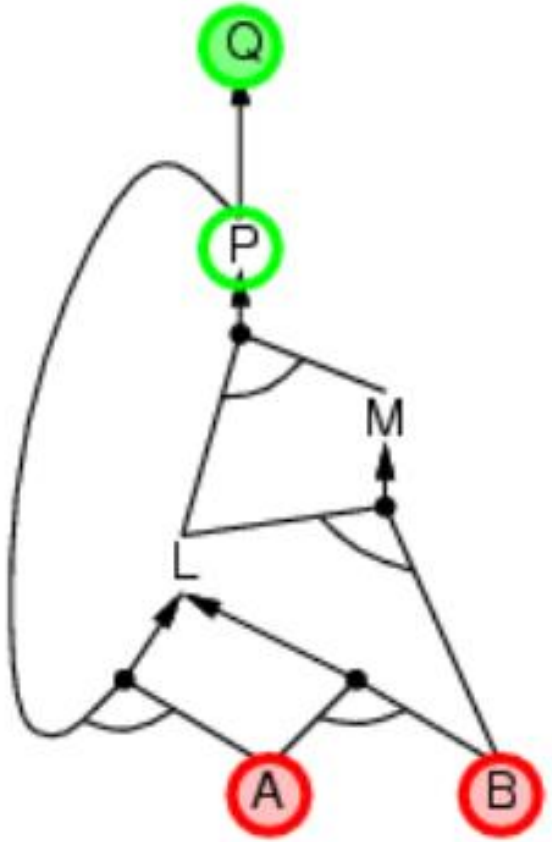
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A ←

B ←



Backward chaining example

Efficient implementation

$P \Rightarrow Q$ ←

$L \wedge M \Rightarrow P$ ←

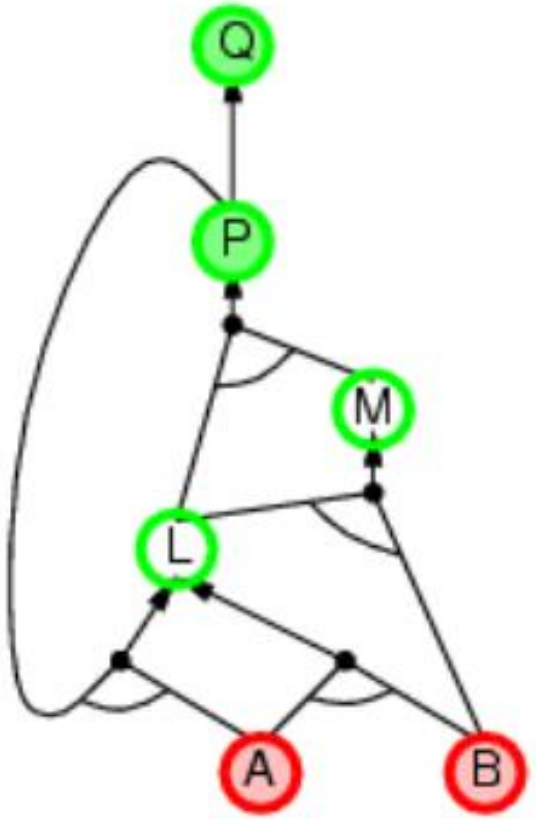
$B \wedge L \Rightarrow M$ ←

$A \wedge P \Rightarrow L$ ←

$A \wedge B \Rightarrow L$ ←

A ←

B ←



Backward chaining example

Efficient implementation

$P \Rightarrow Q$ ←

$L \wedge M \Rightarrow P$ ←

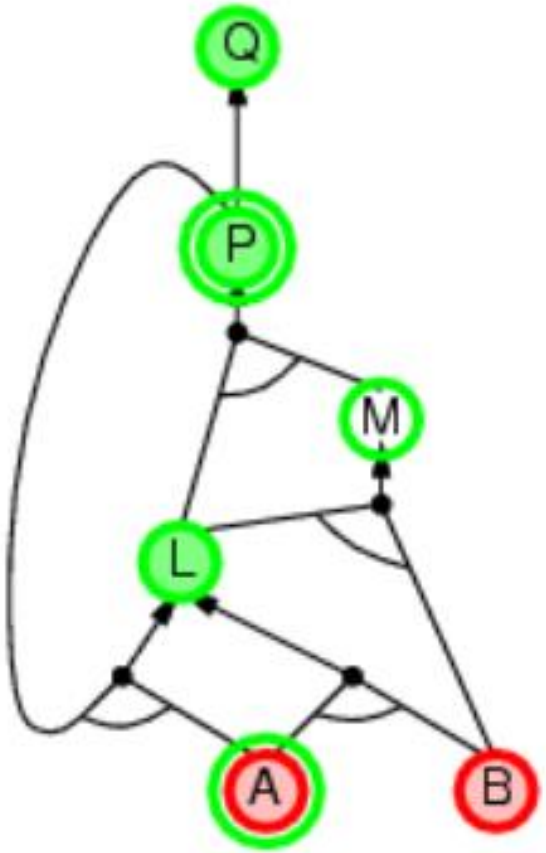
$B \wedge L \Rightarrow M$ ←

$A \wedge P \Rightarrow L$ ←

$A \wedge B \Rightarrow L$ ←

A ←

B ←



Backward chaining example

Efficient implementation

$P \Rightarrow Q$ ←

$L \wedge M \Rightarrow P$ ←

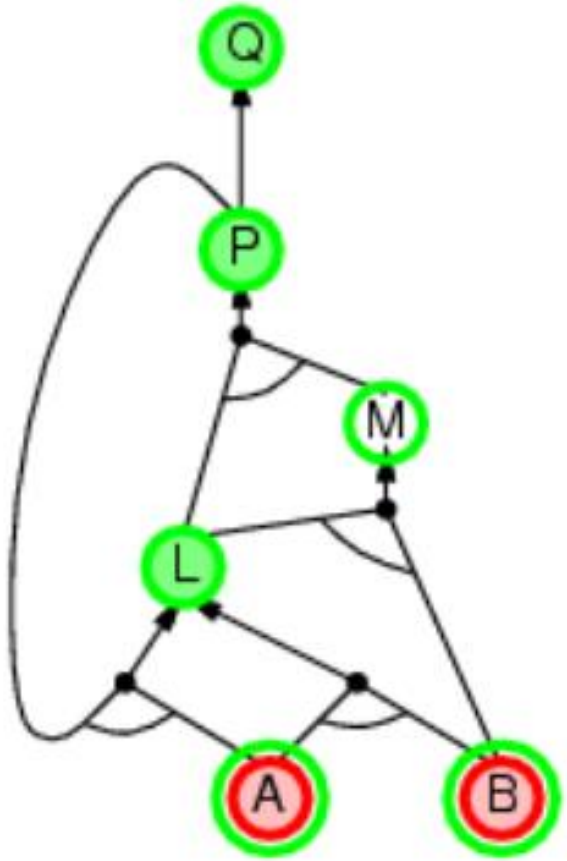
$B \wedge L \Rightarrow M$ ←

$A \wedge P \Rightarrow L$ ←

$A \wedge B \Rightarrow L$ ←

A ←

B ←



Backward chaining example

Efficient implementation

$P \Rightarrow Q$ ←

$L \wedge M \Rightarrow P$ ←

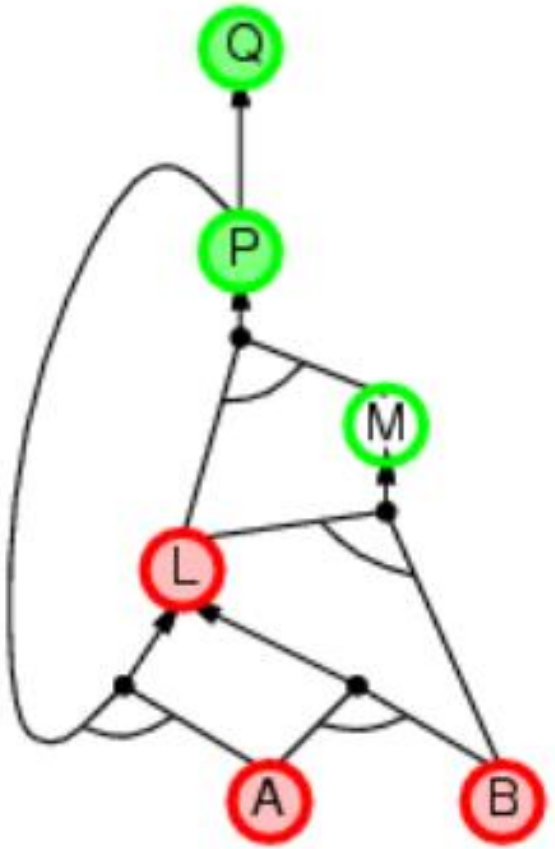
$B \wedge L \Rightarrow M$ ←

$A \wedge P \Rightarrow L$ ←

$A \wedge B \Rightarrow L$ ←

A ←

B ←



Backward chaining example

Efficient implementation

$P \Rightarrow Q$ ←

$L \wedge M \Rightarrow P$ ←

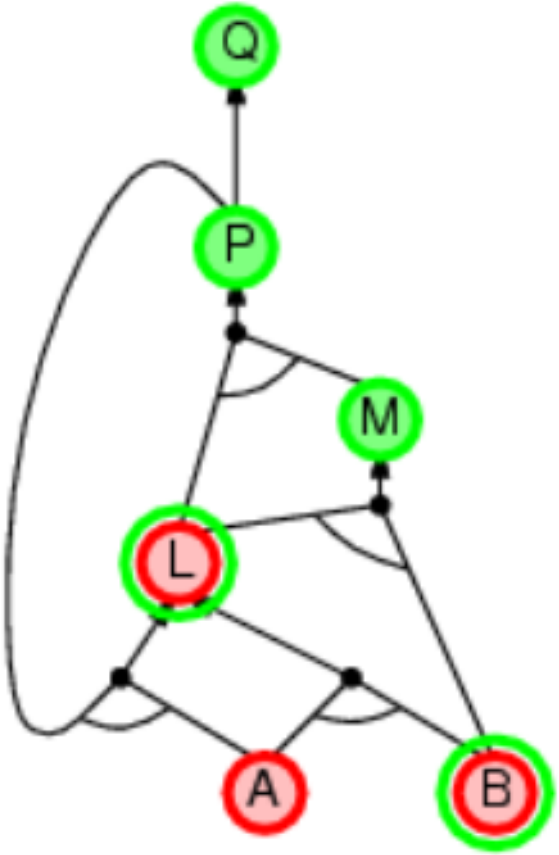
$B \wedge L \Rightarrow M$ ←

$A \wedge P \Rightarrow L$ ←

$A \wedge B \Rightarrow L$ ←

A ←

B ←



Backward chaining example

Efficient implementation

$P \Rightarrow Q$ ←

$L \wedge M \Rightarrow P$ ←

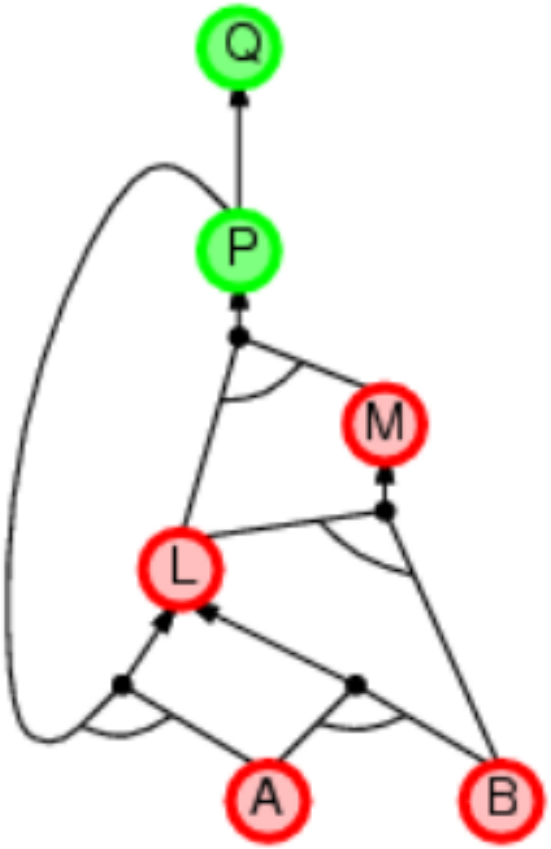
$B \wedge L \Rightarrow M$ ←

$A \wedge P \Rightarrow L$ ←

$A \wedge B \Rightarrow L$ ←

A ←

B ←



Backward chaining example

Efficient implementation

$P \Rightarrow Q$ ←

$L \wedge M \Rightarrow P$ ←

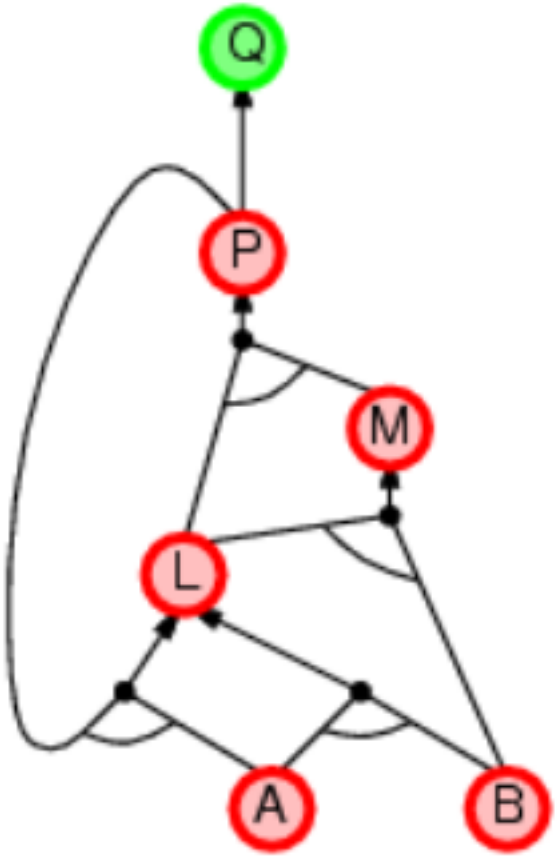
$B \wedge L \Rightarrow M$ ←

$A \wedge P \Rightarrow L$ ←

$A \wedge B \Rightarrow L$ ←

A ←

B ←



Backward chaining example

Efficient implementation

$P \Rightarrow Q$ ←

$L \wedge M \Rightarrow P$ ←

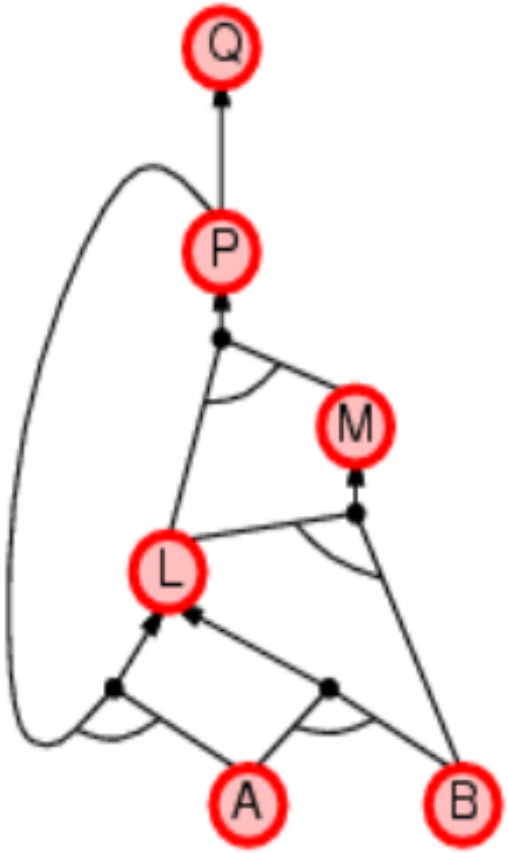
$B \wedge L \Rightarrow M$ ←

$A \wedge P \Rightarrow L$ ←

$A \wedge B \Rightarrow L$ ←

A ←

B ←



FACULTY OF INFORMATION TECHNOLOGY

Thank you for your attention!