

The background of the slide is a dark blue gradient with a complex pattern of glowing blue circuit lines and dots, resembling a digital or neural network. On the left side, there is a dark purple rectangular area containing the text 'AI'.

AI

LOGICAL AGENTS

Nguyễn Ngọc Thảo – Nguyễn Hải Minh
{nnthao, nhminh}@fit.hcmus.edu.vn

Outline

- Knowledge-based agents
- The Wumpus world
- Inference with Propositional logic
- Propositional theorem proving
- Effective propositional model checking

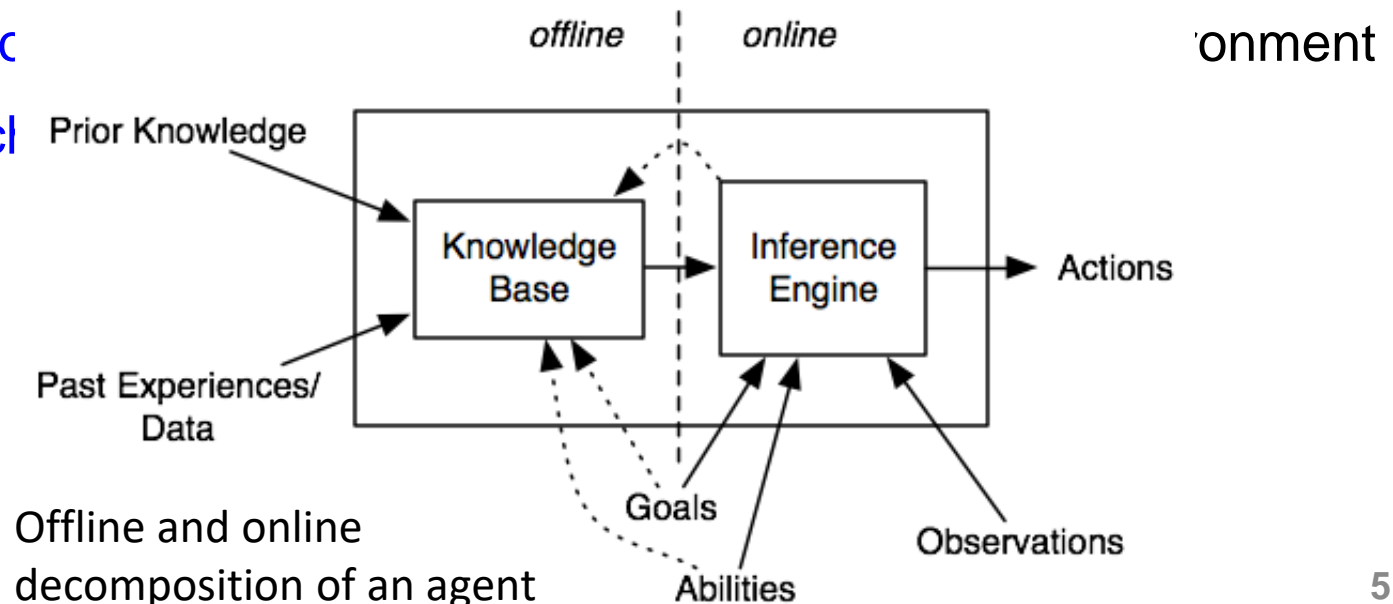
Knowledge-based agents

Problem-solving agents

- These agents **know things in a very limited, inflexible sense**.
 - They know what actions are available and the results of these actions, but they do not know general facts.
 - E.g., an 8-puzzle agent doesn't know that two tiles cannot occupy the same space.
- The **factored representation** in CSP enables some parts of the agent to **work domain-independently**.

Knowledge-based (KB) agents

- Knowledge-based agents **perform reasoning over an internal representation of knowledge** to **decide what actions to take**.
- Supported by **logic** – a general class of representation
- **Combine and recombine information** to suit myriad purposes
 - **Accept new tasks** in the form of explicitly described goals
 - **Achieve cc**
 - **Adapt to cl**



Knowledge-based agents



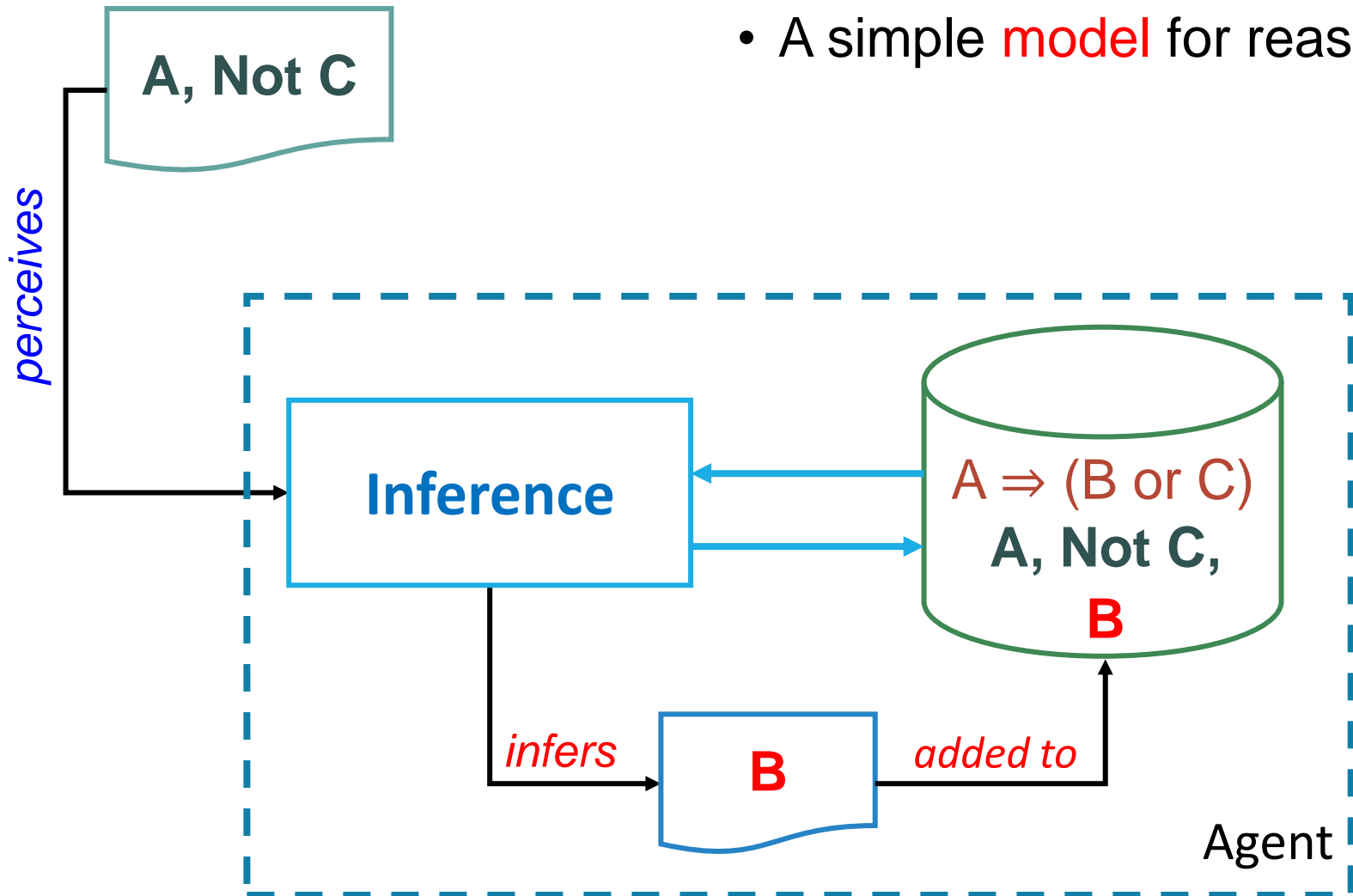
- **Knowledge base (KB):** A set of sentences or facts
 - Each sentence represents some assertion about the world.
 - **Axiom** = sentence that is not derived from other sentences

- **Inference:** Derive (infer) new sentences from old ones
 - Add new sentences to the knowledge base and query what is known



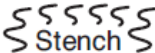


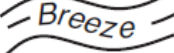
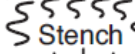


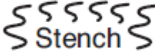
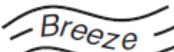

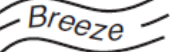
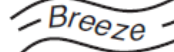
Model for reasoning: An example

- A simple **model** for reasoning



The Wumpus world



4	 Stench		 Breeze	PIT
3		  Stench  Gold	PIT	 Breeze
2	 Stench		 Breeze	
1	 START	 Breeze	PIT	 Breeze
	1	2	3	4

PEAS Description

- Environment

- 4×4 grid of rooms, agent starts in the square [1,1], facing to the right
- The locations of Gold and Wumpus are random
- Each square can be a pit, with probability 0.2

- Performance measure

- +1000 for climbing out of the cave with gold, -1000 for death
- -1 per step, -10 for using the arrow
- The game ends when agent dies or climbs out of the cave

- Actuators: *Forward, TurnLeft/TurnRight by 90°, Grab, Shoot, Climb*

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

- Percept: *[Stench, Breeze, None, None, None]*

Exploring a Wumpus world

A = *Agent*
B = *Breeze*
G = *Glitter, Gold*
OK = *Safe square*
P = *Pit*
S = *Stench*
V = *Visited*
W = *Wumpus*

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

Exploring a Wumpus world

A = Agent
B = Breeze
G = Glitter, Gold
OK = Safe square
P = Pit
S = Stench
V = Visited
W = Wumpus

1,4	2,4	3,4	4,4
1,3	2,3	3,3	4,3
1,2 OK	2,2 P?	3,2	4,2
1,1 V OK	2,1 A B OK	3,1 P?	4,1

Exploring a Wumpus world

A = Agent
B = Breeze
G = Glitter, Gold
OK = Safe square
P = Pit
S = Stench
V = Visited
W = Wumpus

1,4	2,4	3,4	4,4
1,3 W!	2,3	3,3	4,3
1,2 A S OK	2,2 OK	3,2	4,2
1,1 V OK	2,1 B V OK	3,1 P!	4,1

Exploring a Wumpus world

A = Agent
B = Breeze
G = Glitter, Gold
OK = Safe square
P = Pit
S = Stench
V = Visited
W = Wumpus

1,4	2,4 P?	3,4	4,4
1,3 W!	2,3 A S G B	3,3 P?	4,3
1,2 S V OK	2,2 V OK	3,2	4,2
1,1 V OK	2,1 B V OK	3,1 P!	4,1

An agent in the Wumpus world

- A Wumpus-world agent using propositional logic will have a KB of **64** distinct **proposition symbols**, **155** sentences.

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

$$\neg W_{1,1}$$

$$B_{x,y} \Leftrightarrow (P_{x,y+1} \vee P_{x,y-1} \vee P_{x+1,y} \vee P_{x-1,y})$$

$$S_{x,y} \Leftrightarrow (W_{x,y+1} \vee W_{x,y-1} \vee W_{x+1,y} \vee W_{x-1,y})$$

$$W_{1,1} \vee W_{1,2} \vee \dots \vee W_{4,4}$$

$$\neg W_{1,1} \vee \neg W_{1,2}$$

$$\neg W_{1,1} \vee \neg W_{1,3}$$

...

Propositional logic

Propositional logic: Syntax

$$\begin{aligned} \textit{Sentence} &\rightarrow \textit{AtomicSentence} \mid \textit{ComplexSentence} \\ \textit{AtomicSentence} &\rightarrow \textit{True} \mid \textit{False} \mid P \mid Q \mid R \mid \dots \\ \textit{ComplexSentence} &\rightarrow (\textit{Sentence}) \mid [\textit{Sentence}] \\ &\mid \neg \textit{Sentence} \\ &\mid \textit{Sentence} \wedge \textit{Sentence} \\ &\mid \textit{Sentence} \vee \textit{Sentence} \\ &\mid \textit{Sentence} \Rightarrow \textit{Sentence} \\ &\mid \textit{Sentence} \Leftrightarrow \textit{Sentence} \end{aligned}$$

OPERATOR PRECEDENCE : $\neg, \wedge, \vee, \Rightarrow, \Leftrightarrow$

Literal: atomic sentence (P) or negated atomic sentence ($\neg P$)

Logics in general

- **Models** (or possible worlds) are mathematical abstractions that fix the truth or falsehood of every relevant sentence.
 - E.g., all possible assignments of real numbers to x and y
- m **satisfies** (or **is a model of**) α if α is true in model m
- $M(\alpha)$ = the set of all models of α

Propositional logic: Semantics

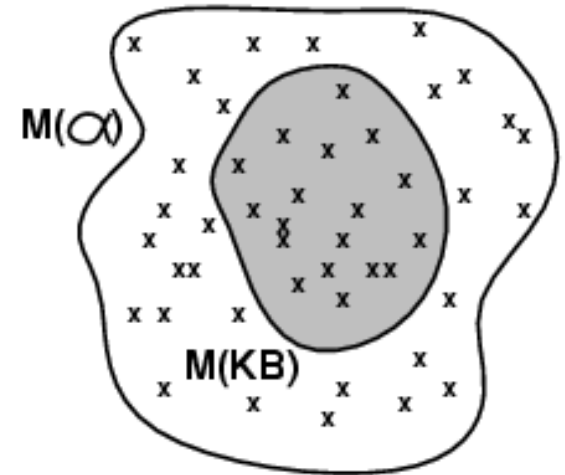
- Each model specifies true/false for each proposition symbol.
 - E.g., $m_1 = \{P_{1,2} = \text{false}, P_{2,2} = \text{false}, P_{3,1} = \text{true}\}$, 8 possible models
- Rules for evaluating truth with respect to a model m

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

- Simple recursive process evaluates an arbitrary sentence.
 - E.g., $\neg P_{1,2} \wedge (P_{2,2} \vee P_{3,1}) = \text{true} \wedge (\text{true} \vee \text{false}) = \text{true} \wedge \text{true} = \text{true}$

Entailment in logic

- A sentence **follows logically** from another sentence: $\alpha \models \beta$
- $\alpha \models \beta$ if and only if, in every model in which α is true, β is also true, i.e., $M(\alpha) \subseteq M(\beta)$
- For example,
 - $x = 0$ entails $xy = 0$
 - The KB containing “Apple is red” and “Tomato is red” entails “Either the apple or the tomato is red”
- Entailment is a relationship between sentences (i.e., **syntax**) that is based on **semantics**.

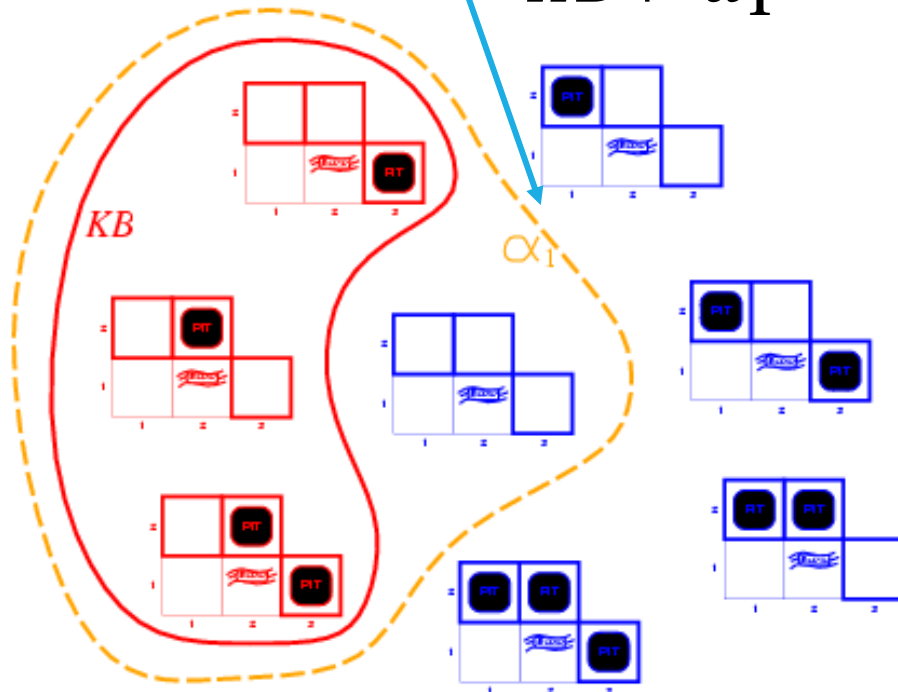


Entailment in logic: Wumpus world

- Consider two possible conclusions α_1 and α_2

“There is no pit in [1,2].”

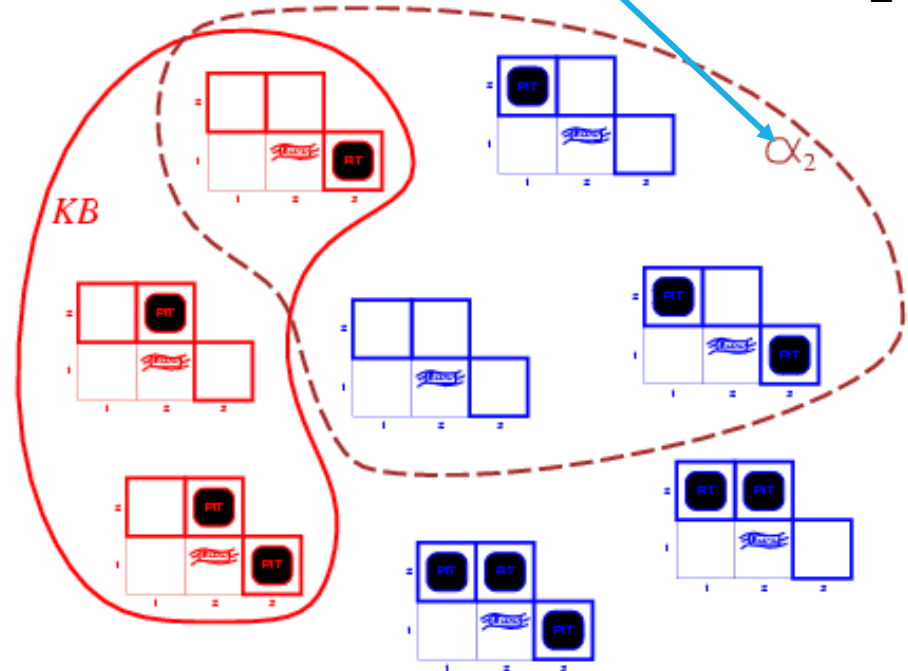
$KB \models \alpha_1$



(a)

“There is no pit in [2,2].”

$KB \not\models \alpha_2$

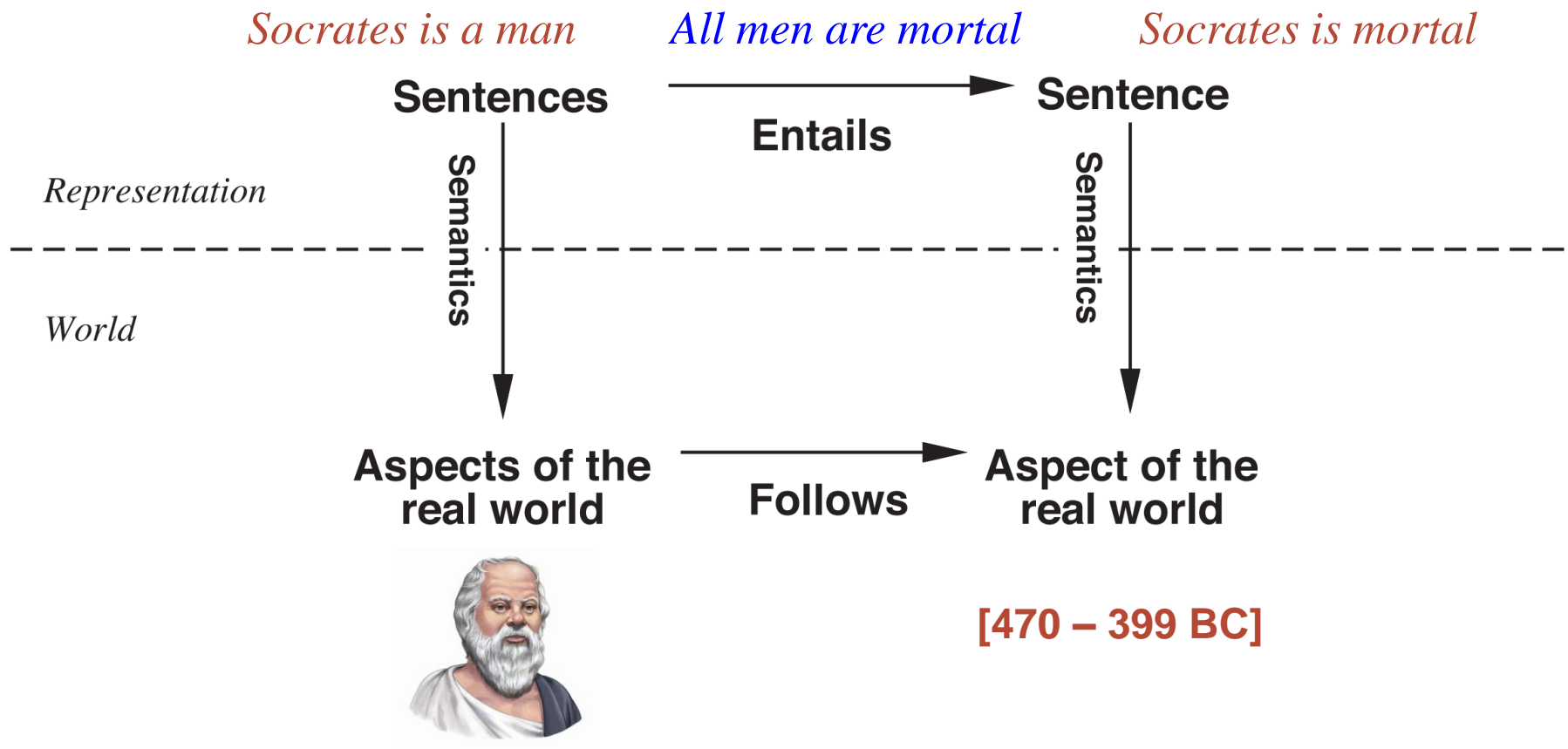


(b)

Logical inference

- Let $KB \models_i \alpha$ mean α can be derived from KB by procedure i .
- Procedure i is sound if whenever $KB \models_i \alpha$, it is also true that $KB \models \alpha$
- Procedure i is complete if whenever $KB \models \alpha$, it is also true that $KB \models_i \alpha$
- That is, the procedure will answer any question whose answer follows from what is known by the KB .

World and representation



A simple knowledge base

- Symbols for each position $[i, j]$
 - $P_{i,j}$: there is a pit in $[i, j]$
 - $W_{i,j}$: there is a Wumpus in $[i, j]$
 - $B_{i,j}$: there is a breeze in $[i, j]$
 - $S_{i,j}$: there is a stench in $[i, j]$
- Sentences in Wumpus world's KB

$$R_1: \neg P_{1,1}$$

$$R_2: B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$$

$$R_3: B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$$

$$R_4: \neg B_{1,1}$$

$$R_5: B_{2,1}$$

1,4	2,4	3,4	4,4
1,3	2,3	3,3	4,3
1,2 OK	2,2 P?	3,2	4,2
1,1 V OK	2,1 A B OK	3,1 P?	4,1

A simple inference procedure

- **Given:** a set of sentences, KB , and sentence α
- **Goal:** answer $KB \models \alpha?$ = “Does KB semantically entail α ?”
 - In all interpretations in which KB ’s sentences are true, is α also true?
 - E.g., in the Wumpus world, $KB \models P_{1,2}$? = “Is there a pit in [1,2]?”

Model-checking approach (Inference by enumeration)

Inference rules

Conversion to the inverse SAT problem (Resolution refutation)

Model-checking approach

- Check if α is true in every model in which KB is true.
 - E.g., the Wumpus's KB has 7 symbols $\rightarrow 2^7 = 128$ models
- Draw a truth table for checking

No pit in [1,2]

$B_{1,1}$	$B_{2,1}$	$P_{1,1}$	$P_{1,2}$	$P_{2,1}$	$P_{2,2}$	$P_{3,1}$	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
\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots
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	<u>true</u>
false	true	false	false	false	true	false	true	true	true	true	true	<u>true</u>
false	true	false	false	false	true	true	true	true	true	true	true	<u>true</u>
false	true	false	false	true	false	false	true	false	false	true	true	false
\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots
true	true	true	true	true	true	true	false	true	true	false	true	false

Inference by (depth-first) enumeration

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

$symbols \leftarrow$ a list of the proposition symbols in KB and α

return TT-CHECK-ALL($KB, \alpha, symbols, \{ \}$)

function TT-CHECK-ALL($KB, \alpha, symbols, model$) **returns** *true* or *false*

if EMPTY?($symbols$) **then**

if PL-TRUE?($KB, model$) **then return** PL-TRUE?($\alpha, model$)

else return *true* // when KB is false, always return *true*

else do

$P \leftarrow$ FIRST($symbols$)

$rest \leftarrow$ REST($symbols$)

return (TT-CHECK-ALL($KB, \alpha, rest, model \cup \{P = true\}$)

and TT-CHECK-ALL($KB, \alpha, rest, model \cup \{P = false\}$))

sound and **complete**

Time complexity $O(2^n)$, space complexity $O(n)$

Quiz 01: Model-checking approach

- Given a KB containing the following rules and facts
 - R_1 : IF hot AND smoky THEN fire
 - R_2 : IF alarm_beeps THEN smoky
 - R_3 : IF fire THEN sprinklers_on
 - F_1 : alarm_beeps
 - F_2 : hot
- Represent the KB in propositional logic with given symbols
 - H = hot, S = smoky, F = fire, A = alarms_beeps, R = sprinklers_on
- Answer the question “Sprinklers_on?” by using the model-checking approach.

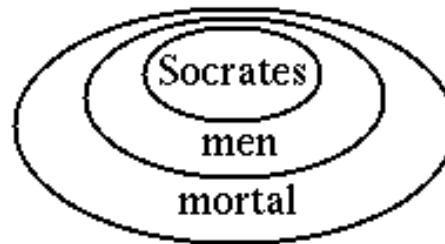


Propositional theorem proving

- *Proof by Resolution*
 - *Forward and Backward Chaining*
-

Inference rules approach

- **Theorem proving:** Apply **rules of inference directly** to the sentences in KB to construct a **proof of the desired sentence without consulting models**
- More efficient than model checking when the number of models is large, yet the length of the proof is short



men \rightarrow mortal
Socrates \rightarrow man
Socrates \rightarrow mortal



Logical equivalence

- Two sentences, α and β , are **logically equivalent** if they are true in the same set of models.

$$\alpha \equiv \beta \text{ iff } \alpha \models \beta \text{ and } \beta \models \alpha$$

$(\alpha \wedge \beta) \equiv (\beta \wedge \alpha)$	commutativity of \wedge
$(\alpha \vee \beta) \equiv (\beta \vee \alpha)$	commutativity of \vee
$((\alpha \wedge \beta) \wedge \gamma) \equiv (\alpha \wedge (\beta \wedge \gamma))$	associativity of \wedge
$((\alpha \vee \beta) \vee \gamma) \equiv (\alpha \vee (\beta \vee \gamma))$	associativity of \vee
$\neg(\neg\alpha) \equiv \alpha$	double-negation elimination
$(\alpha \Rightarrow \beta) \equiv (\neg\beta \Rightarrow \neg\alpha)$	contraposition
$(\alpha \Rightarrow \beta) \equiv (\neg\alpha \vee \beta)$	implication elimination
$(\alpha \Leftrightarrow \beta) \equiv ((\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha))$	biconditional elimination
$\neg(\alpha \wedge \beta) \equiv (\neg\alpha \vee \neg\beta)$	De Morgan
$\neg(\alpha \vee \beta) \equiv (\neg\alpha \wedge \neg\beta)$	De Morgan
$(\alpha \wedge (\beta \vee \gamma)) \equiv ((\alpha \wedge \beta) \vee (\alpha \wedge \gamma))$	distributivity of \wedge over \vee
$(\alpha \vee (\beta \wedge \gamma)) \equiv ((\alpha \vee \beta) \wedge (\alpha \vee \gamma))$	distributivity of \vee over \wedge

Validity

- A sentence is **valid** if it is true in all models.
 - E.g., $P \vee \neg P$, $(P \wedge (P \Rightarrow Q)) \Rightarrow Q$
- Valid sentences are also known as **tautologies**.
- Validity is connected to inference via the **Deduction Theorem**
 $\alpha \models \beta$ iff $\alpha \Rightarrow \beta$ is valid

Satisfiability

- A sentence is **satisfiable** if it is true in **some** model.
 - E.g., $P \vee Q$, P
- A sentence is **unsatisfiable** if it is true in **no** models.
 - E.g., $P \wedge \neg P$
- Satisfiability is connected to inference via the following
$$\alpha \models \beta \text{ iff } \alpha \wedge \neg \beta \text{ is unsatisfiable}$$

→ **Refutation** or **proof by contradiction**
- The **SAT problem** determines the satisfiability of sentences in propositional logic (NP-complete)
 - E.g., in CSPs, the constraints are satisfiable by some assignment.

Quiz 02: Validity and Satisfiability

- Check the validity and satisfiability of the below sentences using the truth table

1. $A \vee B \Rightarrow A \wedge C$

2. $A \wedge B \Rightarrow A \vee C$

3. $(A \vee B) \wedge (\neg B \vee C) \Rightarrow A \vee C$

4. $(A \vee \neg B) \Rightarrow A \wedge B$

Inference and Proofs

- **Proof:** A chain of conclusions leads to the desired goal
- Example sound rules of inference

$$\frac{\alpha \Rightarrow \beta \quad \alpha}{\therefore \beta}$$

Modus Ponens

$$\frac{\alpha \Rightarrow \beta \quad \neg \beta}{\therefore \neg \alpha}$$

Modus Tollens

$$\frac{\alpha \quad \beta}{\therefore \alpha \wedge \beta}$$

AND-Introduction

$$\frac{\alpha \wedge \beta}{\therefore \alpha}$$

AND-Elimination

Inference rules: An example

KB	No.	Sentences	Explanation
$P \wedge Q$	1	$P \wedge Q$	From KB
$P \Rightarrow R$	2	$P \Rightarrow R$	From KB
$Q \wedge R \Rightarrow S$	3	$Q \wedge R \Rightarrow S$	From KB
$S?$	4	P	1 And-Elim
	5	R	4,2 Modus Ponens
	6	Q	1 And-Elim
	7	$Q \wedge R$	5,6 And-Intro
	8	S	3,7 Modus Ponens

Inference rules in Wumpus world

$$R_1: \neg P_{1,1}$$

$$R_2: B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$$

$$R_3: B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$$

$$R_4: \neg B_{1,1}$$

$$R_5: B_{2,1}$$

$$\text{Proof: } \neg P_{1,2}$$

- Bi-conditional elimination to $R_2: R_6: (B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})$
- And-Elimination to $R_6: R_7: (P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1}$
- Logical equivalence for contrapositives: $R_8: \neg B_{1,1} \Rightarrow \neg(P_{1,2} \vee P_{2,1})$
- Modus Ponens with R_8 and the percept $R_4: R_9: \neg(P_{1,2} \vee P_{2,1})$
- De Morgan's rule: $R_{10}: \neg P_{1,2} \wedge \neg P_{2,1}$

Proving by search

- Search algorithms can be applied to find a sequence of steps that constitutes a proof.
 - INITIAL STATE: the initial knowledge base
 - ACTIONS: apply all inference rules to all the sentences that match the top half of the inference rule
 - RESULT: add the sentence in the bottom half of the inference rule
 - GOAL: a state that contains the sentence need to be proved
- The proof can ignore irrelevant propositions, no matter how many of them there are → **more efficient**
 - E.g., in the Wumpus world, $B_{2,1}$, $P_{1,1}$, $P_{2,2}$ and $P_{3,1}$ are not mentioned.

Monotonicity

- The set of entailed sentences only increases as information is added to the knowledge base.

$$\textit{if } KB \models \alpha \textit{ then } KB \wedge \beta \models \alpha$$

- Additional conclusions can be drawn without invalidating any conclusion α already inferred.

Proof by Resolution

- **Proof by Inference Rules:** sound but not complete
 - If the rules are inadequate, then the goal is not reachable.
- **Resolution:** sound and complete, a single inference rule
 - A **complete** inference algorithm when coupled with any complete search algorithm

- Unit resolution inference rule

where l_i and m are **complementary literals**

$$\frac{l_1 \vee \cdots \vee l_k \quad m}{l_1 \vee \cdots \vee l_{i-1} \vee l_{i+1} \vee \cdots \vee l_k}$$

- Full resolution rule

$$\frac{l_1 \vee \cdots \vee l_k \quad m_1 \vee \cdots \vee m_n}{l_1 \vee \cdots \vee l_{i-1} \vee l_{i+1} \vee \cdots \vee l_k \vee m_1 \vee \cdots \vee m_{j-1} \vee m_{j+1} \vee \cdots \vee m_n}$$

where l_i and m_j are complementary literals

Inference rules in Wumpus world

$$R_1: \neg P_{1,1}$$

$$R_2: B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$$

$$R_3: B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$$

$$R_4: \neg B_{1,1}$$

$$R_5: B_{2,1}$$

$$R_6: \left(B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1}) \right) \wedge \left((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1} \right)$$

$$R_7: \neg P_{1,2} \wedge \neg P_{2,1} \Rightarrow B_{1,1}$$

$$R_8: \neg B_{1,1} \Rightarrow \neg(P_{1,2} \vee P_{2,1})$$

$$R_9: \neg(P_{1,2} \vee P_{2,1})$$

$$R_{10}: \neg P_{1,2} \wedge \neg P_{2,1}$$

1,4	2,4	3,4	4,4
1,3	2,3	3,3	4,3
1,2	2,2 P?	3,2	4,2
OK			
1,1 V OK	2,1 A B OK	3,1 P?	4,1

Inference rules in Wumpus world

$R_1: \neg P_{1,1}$

...

$R_{11}: \neg B_{1,2}$

$R_{12}: B_{1,2} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{1,3})$

$R_{13}: \neg P_{2,2}$

$R_{14}: \neg P_{1,3}$

$R_{15}: P_{1,1} \vee P_{2,2} \vee P_{3,1}$

$R_{16}: P_{1,1} \vee P_{3,1}$

$R_{17}: P_{3,1}$

1,4	2,4	3,4	4,4
1,3 W!	2,3	3,3	4,3
1,2 A S OK	2,2 OK	3,2	4,2
1,1 V OK	2,1 B V OK	3,1 P!	4,1

$\neg P_{2,2}$ resolves with $P_{2,2}$

$\neg P_{1,1}$ resolves with $P_{1,1}$

Proof by Resolution

- **Factoring:** the resulting clause should contain only one copy of each literal.
 - E.g., resolving $(A \vee B)$ with $(A \vee \neg B)$ obtains $(A \vee A) \rightarrow$ reduced to A
- For *any pair of sentences, α and β , in propositional logic, a resolution-based theorem prover can decide whether $\alpha \models \beta$.*

Conjunctive Normal Form (CNF)

- Resolution applies only to clauses, i.e., disjunctions of literals
→ Convert all sentences in KB into clauses (CNF form)
- For example, convert $B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$ into CNF
$$(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge (\neg P_{1,2} \vee B_{1,1}) \wedge (\neg P_{2,1} \vee B_{1,1})$$

→ A conjunction of 3 clauses

Conversion to CNF

1. Eliminate \Leftrightarrow : $\alpha \Leftrightarrow \beta \equiv (\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)$
2. Eliminate \Rightarrow : $\alpha \Rightarrow \beta \equiv \neg\alpha \vee \beta$
3. The operator \neg appears only in literals: “move \neg inwards”
 - $\neg\neg\alpha \equiv \alpha$ (double-negation elimination)
 - $\neg(\alpha \wedge \beta) \equiv \neg\alpha \vee \neg\beta$ (De Morgan)
 - $\neg(\alpha \vee \beta) \equiv \neg\alpha \wedge \neg\beta$ (De Morgan)
4. Apply the distributivity law to distribute \vee over \wedge
 - $(\alpha \wedge \beta) \vee \gamma \equiv (\alpha \vee \gamma) \wedge (\beta \vee \gamma)$

Quiz 03: Conversion to CNF

- Convert the following sentences into CNF

1. $(A \wedge B) \Rightarrow (C \Rightarrow D)$

2. $P \vee Q \Leftrightarrow R \wedge \neg Q \Rightarrow P$

Resolution algorithm

- **Proof by contradiction** (resolution refutation): To show that $KB \models \alpha$, prove $KB \wedge \neg\alpha$ is **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

$clauses \leftarrow$ the set of clauses in the CNF representation of $KB \wedge \neg\alpha$

$new \leftarrow \{ \}$

loop do

for each pair of clauses C_i, C_j **in** $clauses$ **do**

$resolvents \leftarrow$ PL-RESOLVE(C_i, C_j)

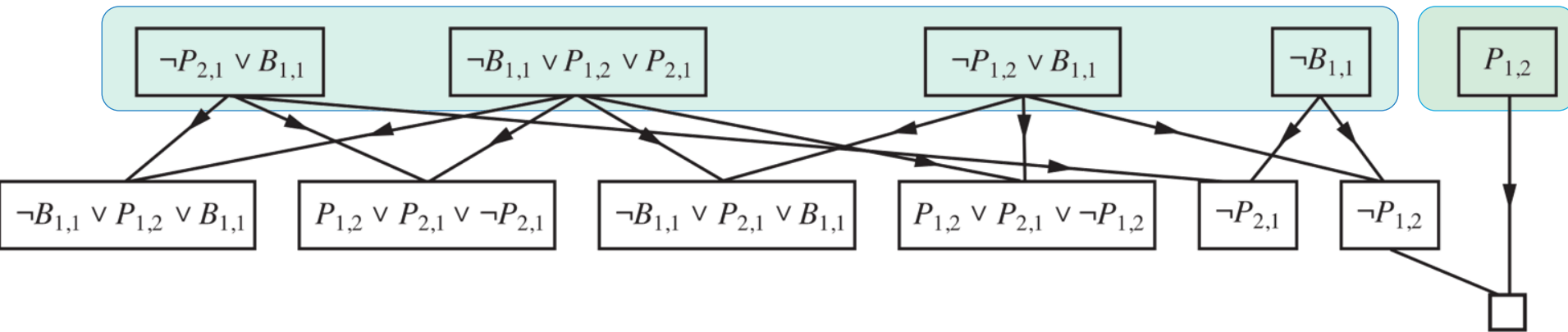
if $resolvents$ contains the empty clause **then return** *true*

$new \leftarrow new \cup resolvents$

if $new \subseteq clauses$ **then return** *false*

$clauses \leftarrow clauses \cup new$

Resolution algorithm



- Many resolution steps are pointless.
- Clauses with two complementary literals can be discarded.
 - E.g., $B_{1,1} \vee \neg B_{1,1} \vee P_{2,1} \equiv \text{True} \vee P_{2,1} \equiv \text{True}$

Quiz 04: The resolution algorithm

- Given the following hypotheses
 - If it rains, Joe brings his umbrella.
 - If Joe brings his umbrella, Joe does not get wet.
 - If it does not rain, Joe does not get wet.
- Prove that Joe does not get wet.

Quiz 04: Resolution algorithm

- The KB contains facts and hypotheses

KB

$$R \Rightarrow U$$

$$U \Rightarrow \neg W$$

$$\neg R \Rightarrow \neg W$$

- Check if the sentence
 $\neg W$ is entailed by KB?

Horn clauses and Definite clauses

- **Definite clause:** a disjunction of literals of which **exactly one is positive**.
 - E.g., $\neg P \vee \neg Q \vee R$ is a definite clause, whereas $\neg P \vee Q \vee R$ is not.
- **Horn clause:** a disjunction of literals of which **at most one is positive**.
 - All definite clauses are Horn clauses
- **Goal clause:** clauses with **no positive literals**
- Horn clauses are closed under resolution
 - Resolving two Horn clauses will get back a Horn clause.

Propositional sentences and clauses

$$CNFSentence \rightarrow Clause_1 \wedge \dots \wedge Clause_n$$

$$Clause \rightarrow Literal_1 \vee \dots \vee Literal_m$$

$$Literal \rightarrow Symbol \mid \neg Symbol$$

$$Symbol \rightarrow P \mid Q \mid R \mid \dots$$

$$HornClauseForm \rightarrow DefiniteClauseForm \mid GoalClauseForm$$

$$DefiniteClauseForm \rightarrow (Symbol_1 \wedge \dots \wedge Symbol_l) \Rightarrow Symbol$$

$$GoalClauseForm \rightarrow (Symbol_1 \wedge \dots \wedge Symbol_l) \Rightarrow False$$

KB of definite clauses

- KB containing only definite clauses are interesting.
- Every definite clause can be written as an implication.
 - Premise (**body**) is a conjunction of positive literals and Conclusion (**head**) is a single positive literal (**fact**) → easier to understand
 - E.g., $\neg P \vee \neg Q \vee R \equiv (P \wedge Q) \Rightarrow R$
- Inference can be done with forward-chaining and backward-chaining algorithms
 - This type of inference is the basis for **logic programming**.
- Deciding entailment can be done in linear time.

KB: Horn clauses vs. CNF clauses

Disjunctions of literals
($l_1 \vee l_2 \vee \dots \vee l_m$)

CNF clauses

Clause 1

\wedge

Clause 2

$\wedge \dots \wedge$

Clause n

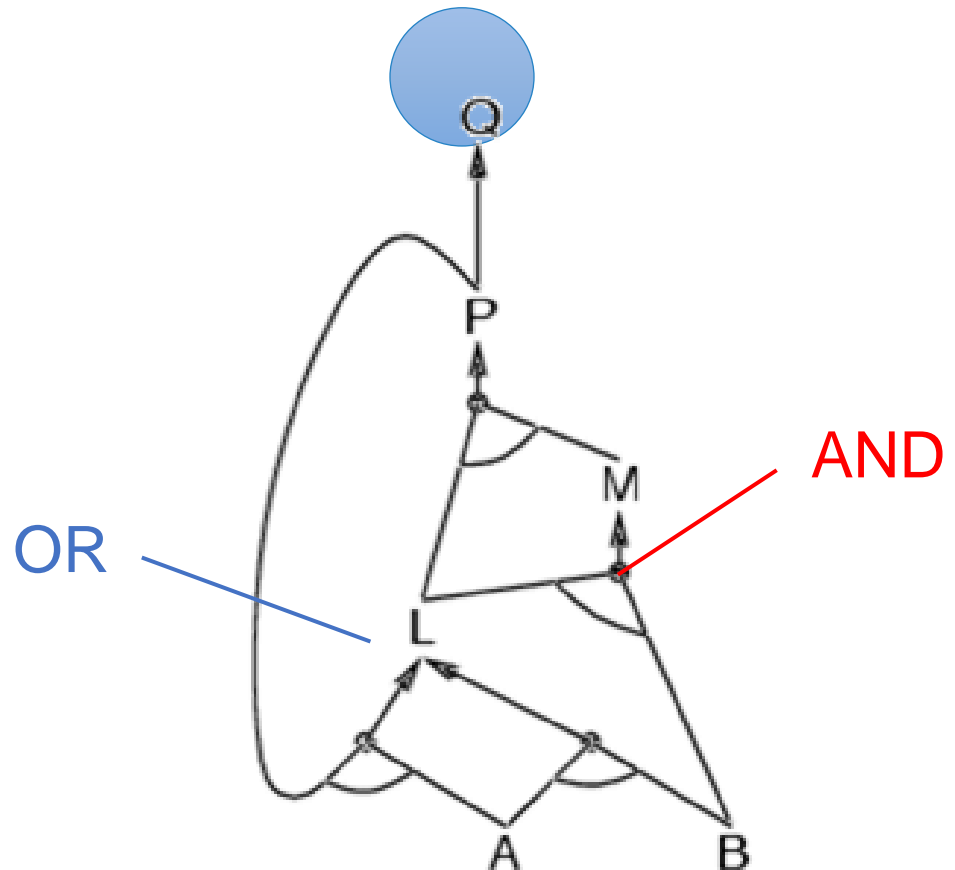
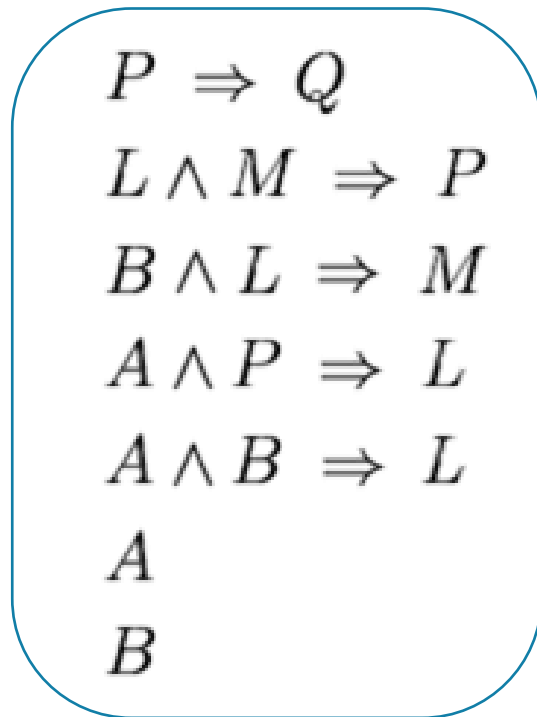
Disjunctions of literals of which **at most one is positive**
($\neg l_1 \vee \neg l_2 \vee \dots \vee l_m$)

Horn clauses

Restricted form

Forward chaining

- **Key idea:** Fire any rule whose premises are satisfied in the KB, add its conclusion to the KB, until the query is found.



The forward chaining algorithm

function PL-FC-ENTAILS?(*KB*, *q*) **returns** *true* or *false*

inputs: *KB*, the knowledge base, a set of propositional definite clauses

q, the query, a proposition symbol

count \leftarrow a table, where *count*[*c*] is the number of symbols in *c*'s premise

inferred \leftarrow a table, where *inferred*[*s*] is initially false for all symbols

agenda \leftarrow a queue of symbols, initially symbols known to be *true* in *KB*

while *agenda* is not empty **do**

p \leftarrow POP(*agenda*)

if *p* = *q* **then return** *true*

if *inferred*[*p*] = *false* **then**

inferred[*p*] \leftarrow *true*

for each clause *c* in *KB* where *p* is in *c*.PREMISE **do**

decrement *count*[*c*]

if *count*[*c*] = 0 **then** add *c*.CONCLUSION to *agenda*

return *false*

Sound and complete

Forward chaining: An example

$$P \Rightarrow Q$$

$$L \wedge M \Rightarrow P$$

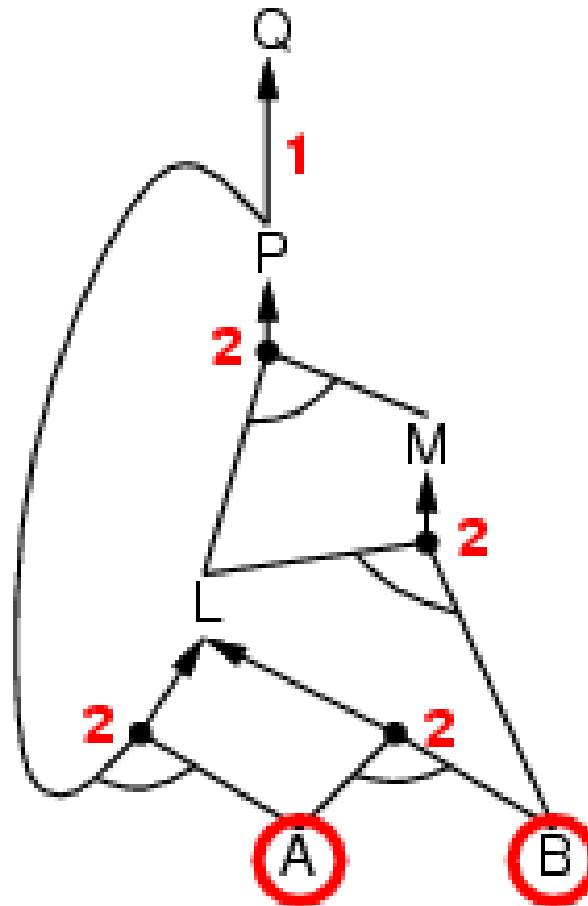
$$B \wedge L \Rightarrow M$$

$$A \wedge P \Rightarrow L$$

$$A \wedge B \Rightarrow L$$

A

B



Forward chaining: An example

$$P \Rightarrow Q$$

$$L \wedge M \Rightarrow P$$

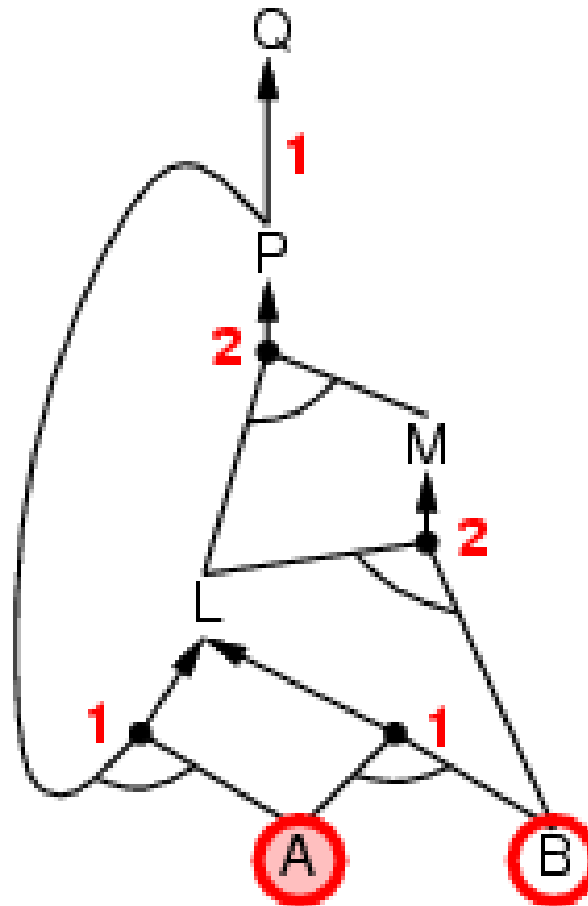
$$B \wedge L \Rightarrow M$$

$$A \wedge P \Rightarrow L$$

$$A \wedge B \Rightarrow L$$

A

B



Forward chaining: An example

$$P \Rightarrow Q$$

$$L \wedge M \Rightarrow P$$

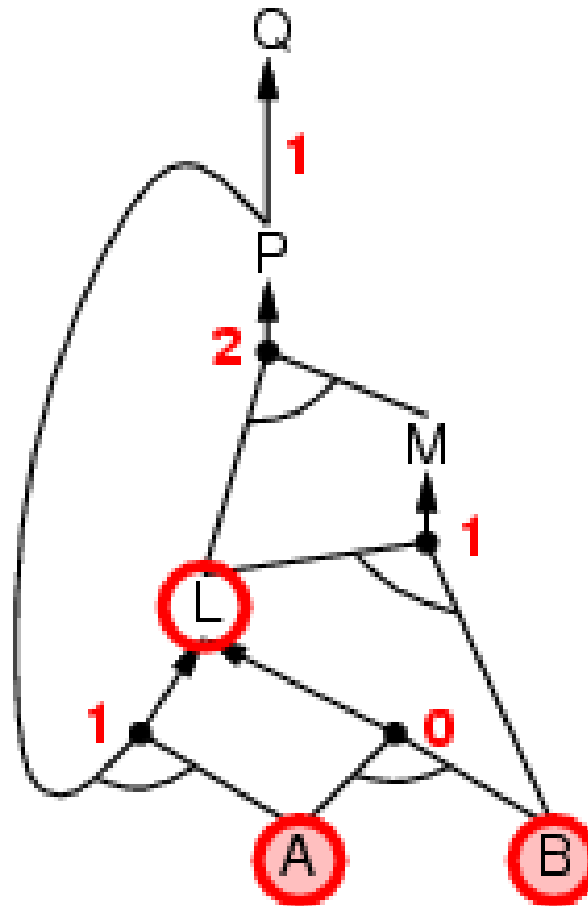
$$B \wedge L \Rightarrow M$$

$$A \wedge P \Rightarrow L$$

$$A \wedge B \Rightarrow L$$

A

B



Forward chaining: An example

$$P \Rightarrow Q$$

$$L \wedge M \Rightarrow P$$

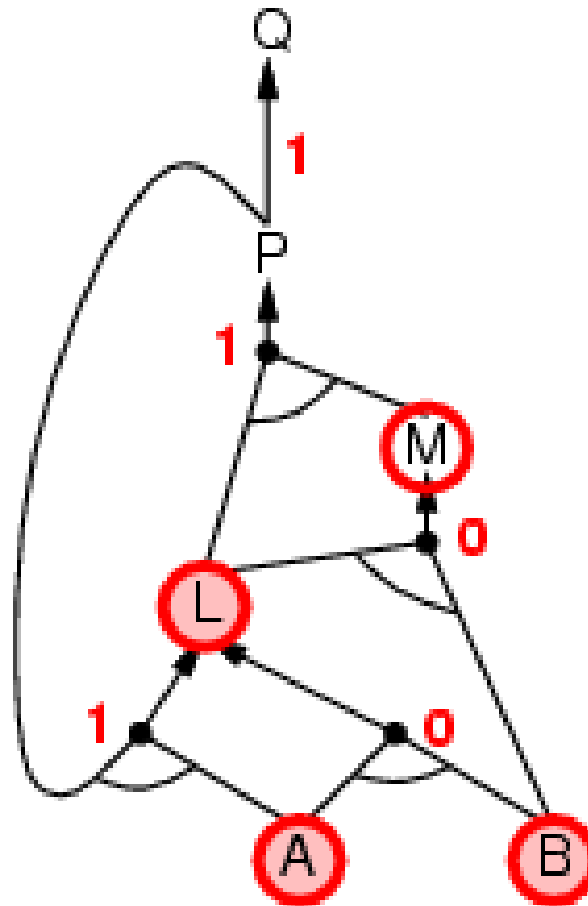
$$B \wedge L \Rightarrow M$$

$$A \wedge P \Rightarrow L$$

$$A \wedge B \Rightarrow L$$

A

B



Forward chaining: An example

$$P \Rightarrow Q$$

$$L \wedge M \Rightarrow P$$

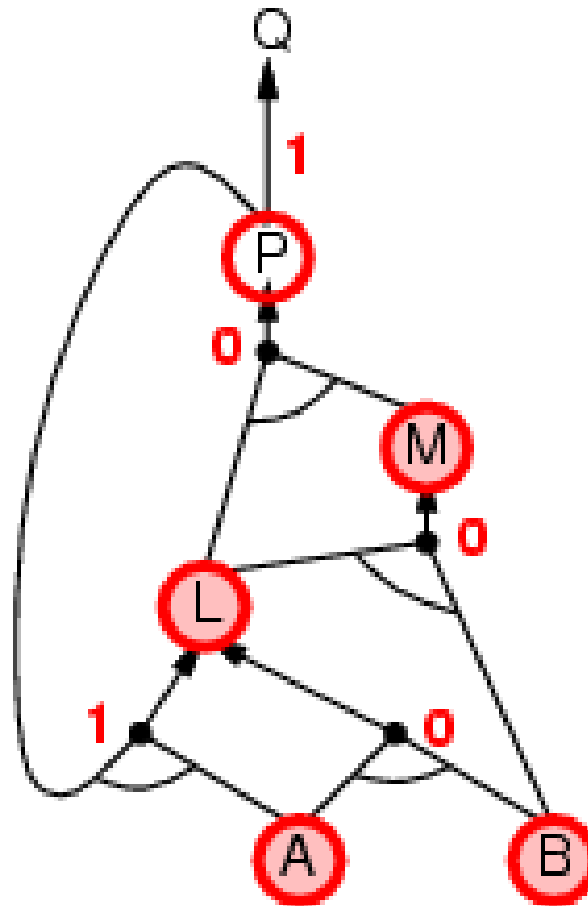
$$B \wedge L \Rightarrow M$$

$$A \wedge P \Rightarrow L$$

$$A \wedge B \Rightarrow L$$

A

B



Forward chaining: An example

$$P \Rightarrow Q$$

$$L \wedge M \Rightarrow P$$

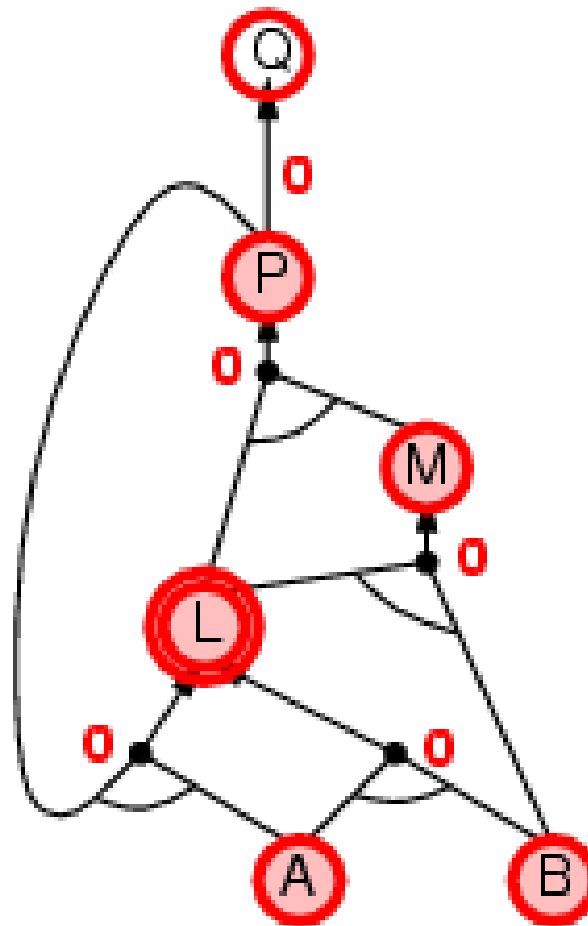
$$B \wedge L \Rightarrow M$$

$$A \wedge P \Rightarrow L$$

$$A \wedge B \Rightarrow L$$

A

B



Forward chaining: An example

$$P \Rightarrow Q$$

$$L \wedge M \Rightarrow P$$

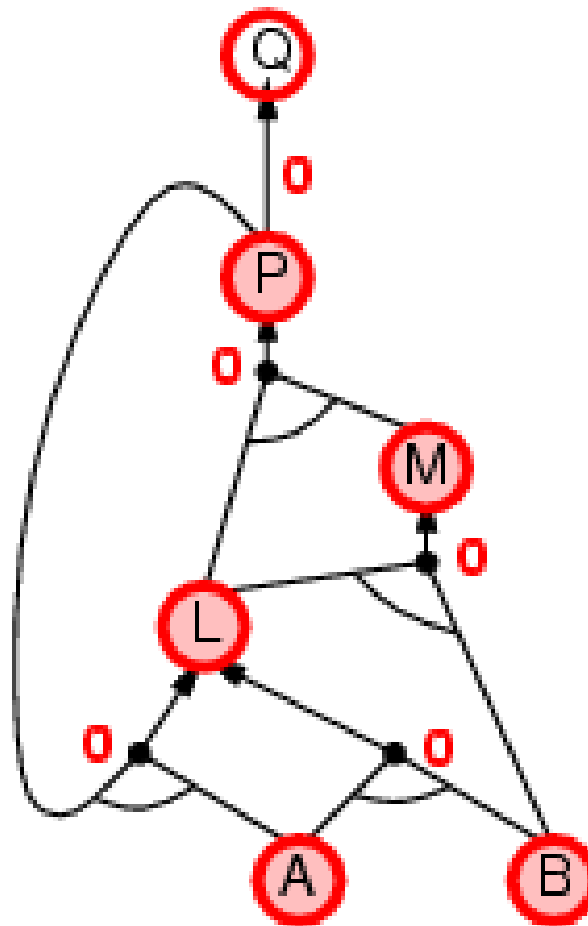
$$B \wedge L \Rightarrow M$$

$$A \wedge P \Rightarrow L$$

$$A \wedge B \Rightarrow L$$

A

B



Forward chaining: An example

$$P \Rightarrow Q$$

$$L \wedge M \Rightarrow P$$

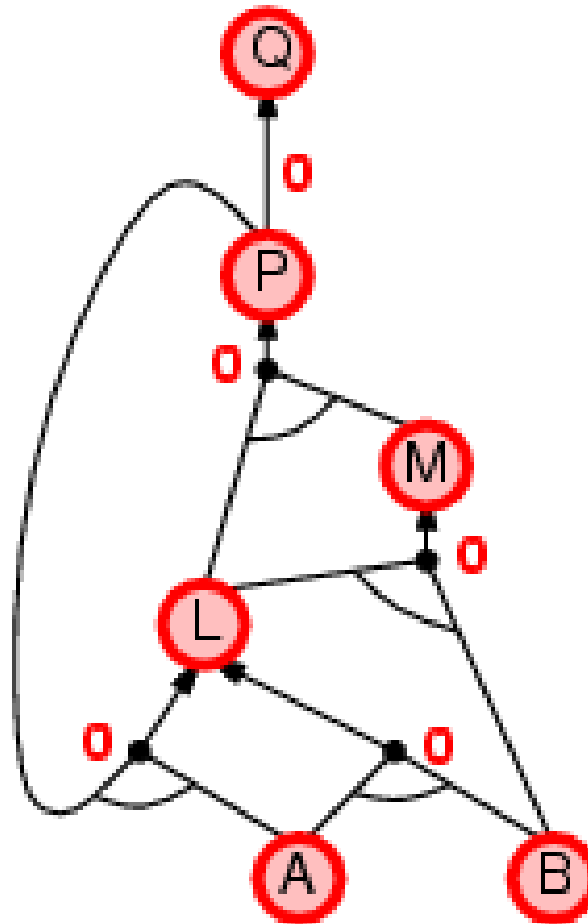
$$B \wedge L \Rightarrow M$$

$$A \wedge P \Rightarrow L$$

$$A \wedge B \Rightarrow L$$

A

B



Forward chaining: Another example

KB	No.	Sentences	Explanation
$A \wedge B \Rightarrow C$	1	$A \wedge B \Rightarrow C$	From KB
$C \wedge D \Rightarrow E$	2	$C \wedge D \Rightarrow E$	From KB
$C \wedge F \Rightarrow G$	3	$C \wedge F \Rightarrow G$	From KB
A	4	A	From KB
B	5	B	From KB
D	6	D	From KB
$E?$	7	C	1, 4 and 5
	8	E	2, 6, and 7

Backward chaining

- **Key idea:** Work backwards from the query q
 - Check if q is known already, or
 - Recursively prove by BC all premises of some rule concluding q
- **Avoid loops:** A new subgoal is already on the goal stack?
- **Avoid repeated work:** A new subgoal has already been proved true, or has already failed?

Backward chaining: An example

$$P \Rightarrow Q$$

$$L \wedge M \Rightarrow P$$

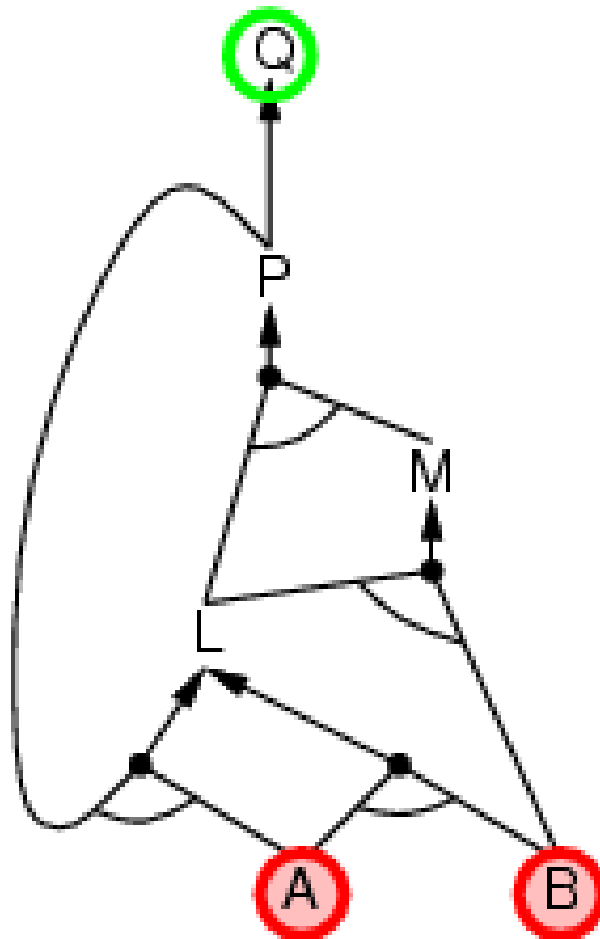
$$B \wedge L \Rightarrow M$$

$$A \wedge P \Rightarrow L$$

$$A \wedge B \Rightarrow L$$

A

B



Backward chaining: An example

$$P \Rightarrow Q$$

$$L \wedge M \Rightarrow P$$

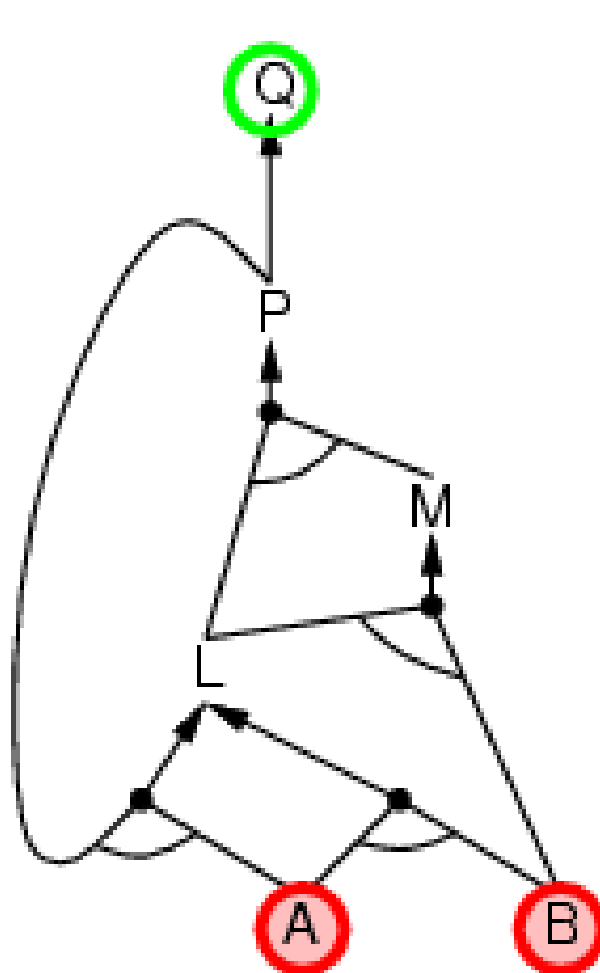
$$B \wedge L \Rightarrow M$$

$$A \wedge P \Rightarrow L$$

$$A \wedge B \Rightarrow L$$

A

B



$Q?$
 $P?$

$P \Rightarrow Q$

Backward chaining: An example

$$P \Rightarrow Q$$

$$L \wedge M \Rightarrow P$$

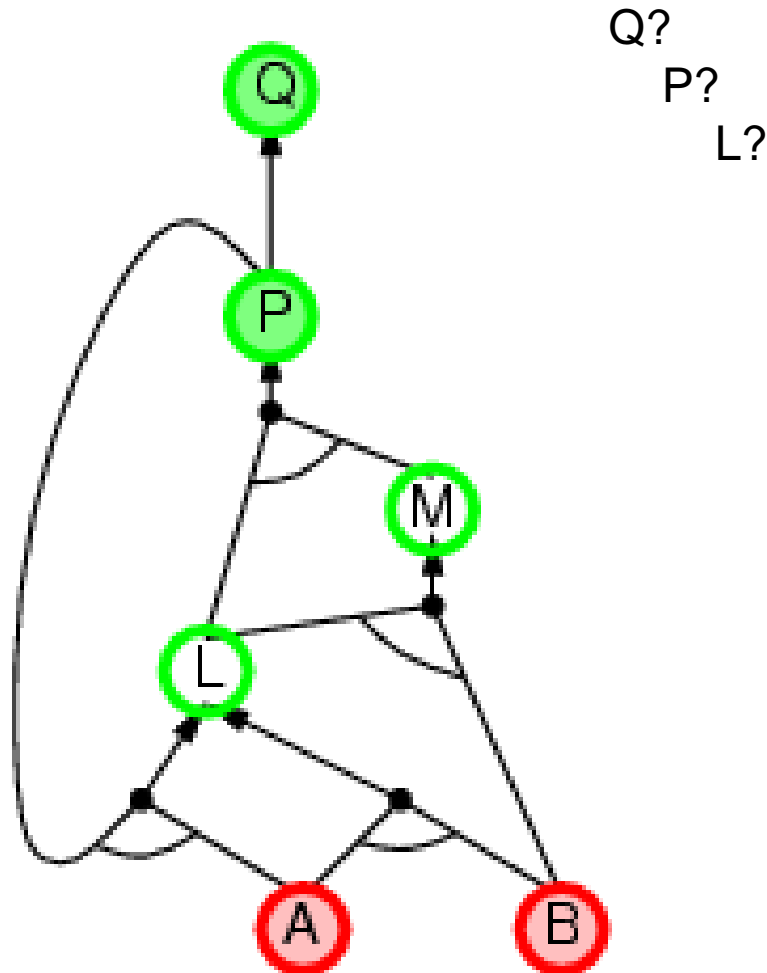
$$B \wedge L \Rightarrow M$$

$$A \wedge P \Rightarrow L$$

$$A \wedge B \Rightarrow L$$

A

B



$$P \Rightarrow Q$$
$$L \wedge M \Rightarrow P$$

Backward chaining: An example

$$P \Rightarrow Q$$

$$L \wedge M \Rightarrow P$$

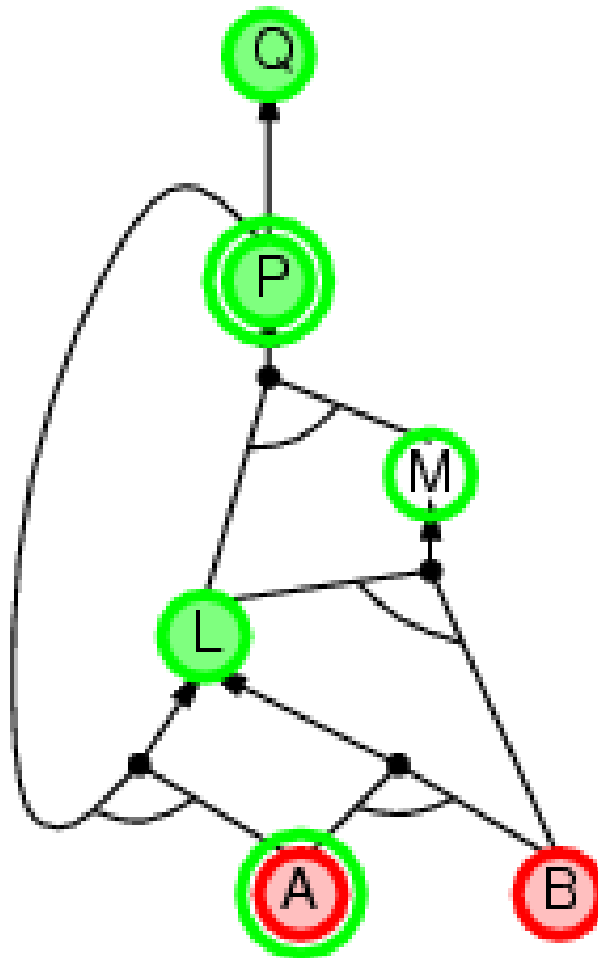
$$B \wedge L \Rightarrow M$$

$$A \wedge P \Rightarrow L$$

$$A \wedge B \Rightarrow L$$

A

B



$Q?$

$P?$

$L?$

$A?$

$P \Rightarrow Q$

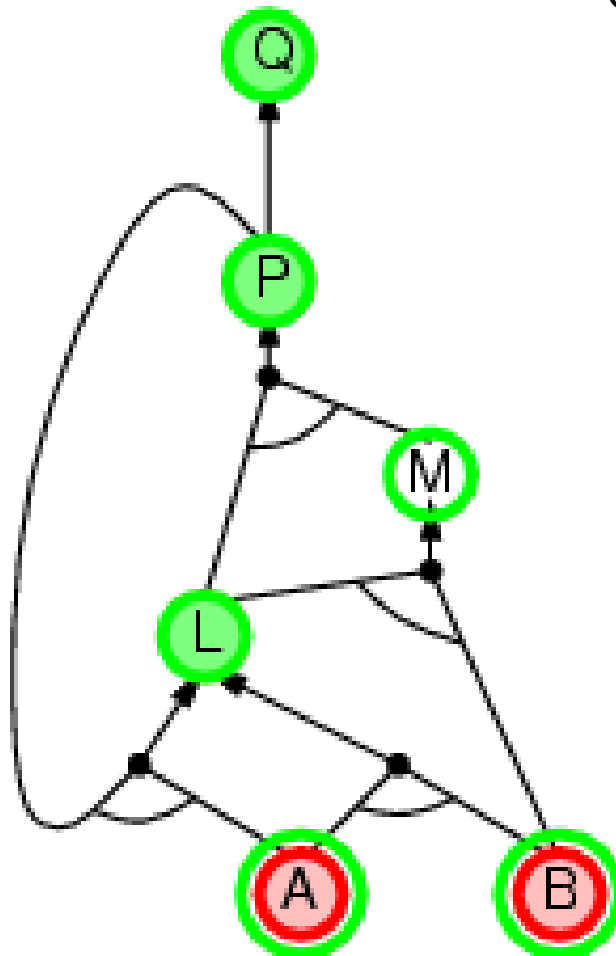
$L \wedge M \Rightarrow P$

$A \wedge B \Rightarrow L$

✓

Backward chaining: An example

$P \Rightarrow Q$
 $L \wedge M \Rightarrow P$
 $B \wedge L \Rightarrow M$
 $A \wedge P \Rightarrow L$
 $A \wedge B \Rightarrow L$
 A
 B

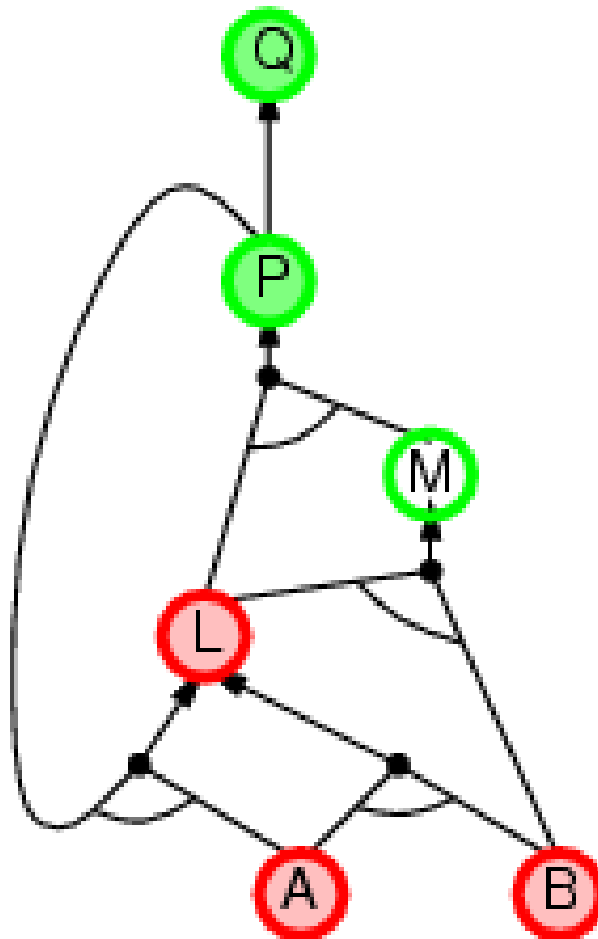


Q?
P?
L?
A?
B?

$P \Rightarrow Q$
 $L \wedge M \Rightarrow P$
 $A \wedge B \Rightarrow L$
✓
✓

Backward chaining: An example

$P \Rightarrow Q$
 $L \wedge M \Rightarrow P$
 $B \wedge L \Rightarrow M$
 $A \wedge P \Rightarrow L$
 $A \wedge B \Rightarrow L$
 A
 B



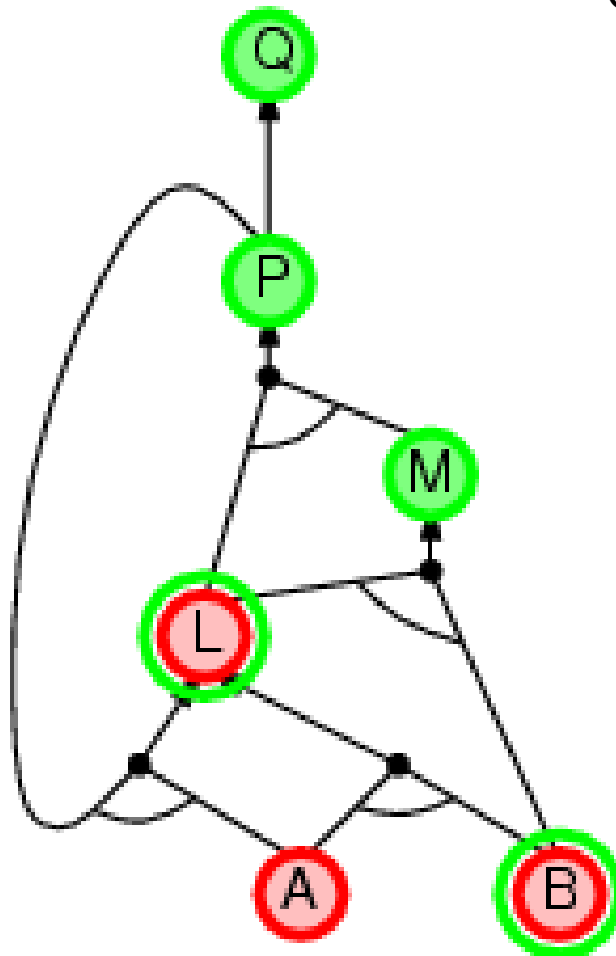
$Q?$
 $P?$
 $L?$ ✓
 $A?$
 $B?$

$P \Rightarrow Q$
 $L \wedge M \Rightarrow P$

 ✓
 ✓

Backward chaining: An example

$P \Rightarrow Q$
 $L \wedge M \Rightarrow P$
 $B \wedge L \Rightarrow M$
 $A \wedge P \Rightarrow L$
 $A \wedge B \Rightarrow L$
 A
 B



Q?
 P?
 L? ✓
 A?
 B?
 M?
 L?
 B?

$P \Rightarrow Q$
 $L \wedge M \Rightarrow P$
 ✓
 ✓
 $L \wedge B \Rightarrow M$

Backward chaining: An example

$$P \Rightarrow Q$$

$$L \wedge M \Rightarrow P$$

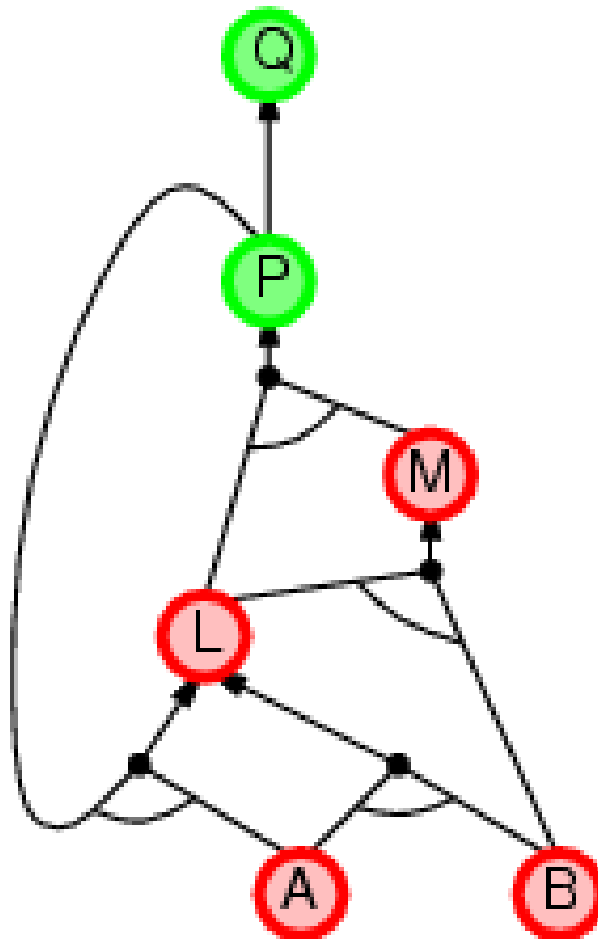
$$B \wedge L \Rightarrow M$$

$$A \wedge P \Rightarrow L$$

$$A \wedge B \Rightarrow L$$

A

B



Q?

P?

L? ✓

A?

B?

M? ✓

L?

B?

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

✓

✓

✓

✓

Backward chaining: An example

$$P \Rightarrow Q$$

$$L \wedge M \Rightarrow P$$

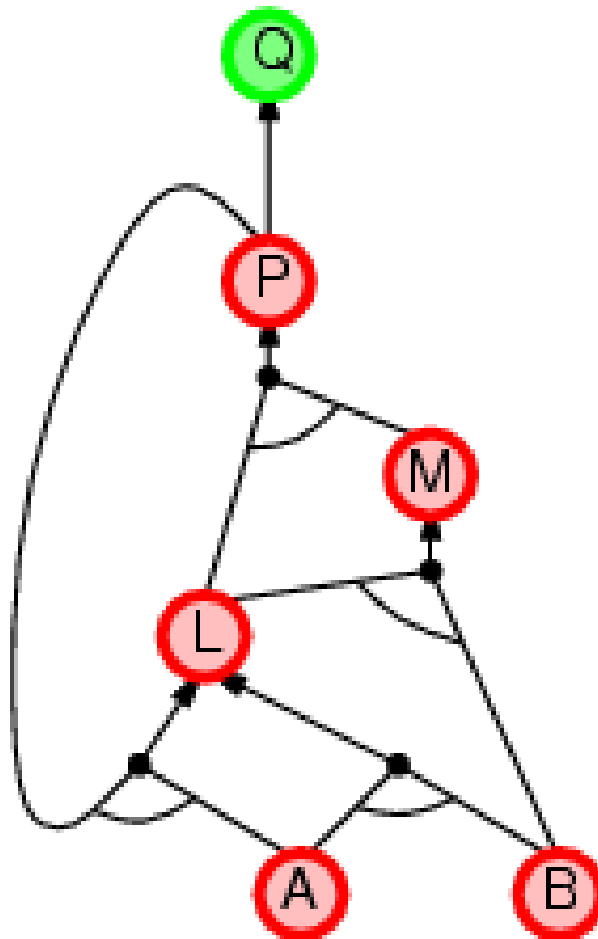
$$B \wedge L \Rightarrow M$$

$$A \wedge P \Rightarrow L$$

$$A \wedge B \Rightarrow L$$

A

B



Q?

✓

P?

✓

L? ✓

A?

✓

B?

✓

M? ✓

L?

✓

B?

✓

Backward chaining: An example

$$P \Rightarrow Q$$

$$L \wedge M \Rightarrow P$$

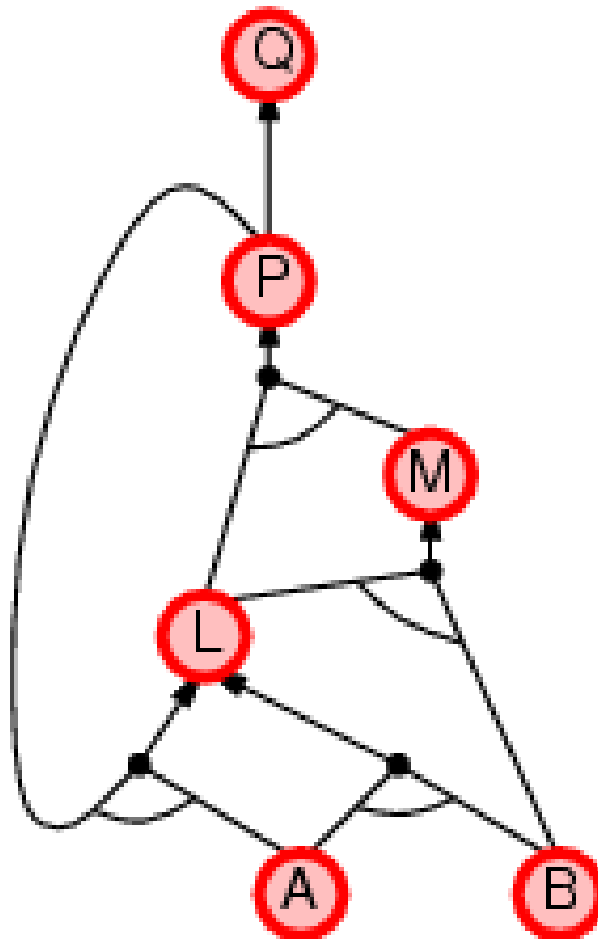
$$B \wedge L \Rightarrow M$$

$$A \wedge P \Rightarrow L$$

$$A \wedge B \Rightarrow L$$

A

B



Q?

✓

P?

✓

L? ✓

A?

✓

B?

✓

M? ✓

L?

✓

B?

✓

Backward chaining: Another example

KB

$A \wedge B \Rightarrow C$

$C \wedge D \Rightarrow E$

$C \wedge F \Rightarrow G$

A

B

D

$E?$

- $E?$

$C \wedge D \Rightarrow E$

- $C?$

$A \wedge B \Rightarrow C$

- $A?$

- $B?$

- $D?$

- A, B and D are given \rightarrow All needed rules are satisfied \rightarrow The goal is proven.

Forward vs. Backward chaining

- **Forward chaining:** **data-driven**, automatic, unconscious processing
 - E.g., object recognition, routine decisions
 - May do lots of work that is irrelevant to the goal
- **Backward chaining:** **goal-driven**, good for problem-solving
 - E.g., Where are my keys? How do I get into a PhD program?
 - Complexity can be **much less** than linear in size of KB

Quiz 05: Forward vs. Backward chaining

- Given a KB containing the following rules and facts
 - R_1 : IF hot AND smoky THEN fire
 - R_2 : IF alarm_beeps THEN smoky
 - R_3 : IF fire THEN sprinklers_on
 - F_1 : alarm_beeps
 - F_2 : hot
- Represent the KB in propositional logic with given symbols
 - H = hot, S = smoky, F = fire, A = alarms_beeps, R = sprinklers_on
- Answer the question “Sprinklers_on?” by using the forward chaining and backward chaining approaches



Effective model checking

- *A complete backtracking algorithm*
 - *Local search algorithms*
-

Efficient propositional inference

- There are two families of efficient algorithms for general propositional inference based on model checking
 1. Complete backtracking search algorithms
 - **DPLL** algorithm (*Davis, Putnam, Logemann, Loveland*)
 2. Incomplete local search algorithms (hill-climbing)
 - **WalkSAT** algorithm

DPLL algorithm

- Often called the **Davis-Putnam algorithm** (1960)
- Determine whether an input propositional logic sentence (in CNF) is satisfiable.
 - A recursive, depth-first enumeration of possible models.
- Improvements over truth table enumeration
 1. Early termination
 2. Pure symbol heuristic
 3. Unit clause heuristic

Improvements in DPLL

- **Early termination:** A clause is **true** if **any literal** is **true**, and a sentence is **false** if **any clause** is **false**.
 - Avoid examination of entire subtrees in the search space
 - E.g., $(A \vee B) \wedge (A \vee C)$ is true if A is true, regardless B and C
- **Pure symbol heuristic:** A **pure symbol** always appears with the same "sign" in all clauses.
 - E.g., $(A \vee \neg B)$, $(\neg B \vee \neg C)$, $(A \vee C)$, A and B are pure, C is impure.
 - Make a pure symbol true \rightarrow Doing so never make a clause false
- **Unit clause heuristic:** there is **only one literal in the clause** and thus this literal **must be true**
 - **Unit propagation:** if the model contains $B = \text{true}$ then $(\neg B \vee \neg C)$ simplifies to a unit clause $\neg C \rightarrow C$ must be false (so that $\neg C$ is true) $\rightarrow A$ must be true (so that $A \vee C$ is true)

DPLL procedure

function DPLL-SATISFIABLE?(*s*) **returns** *true* or *false*
inputs: *s*, a sentence in propositional logic
clauses \leftarrow the set of clauses in the CNF representation of *s*
symbols \leftarrow a list of the proposition symbols in *s*
return DPLL(*clauses*, *symbols*, { })

function DPLL(*clauses*, *symbols*, *model*) **returns** *true* or *false*
if every clause in *clauses* is *true* in *model* **then return** *true*
if some clause in *clauses* is *false* in *model* **then return** *false* } 1. Early Termination
P, *value* \leftarrow ² FIND-PURE-SYMBOL(*symbols*, *clauses*, *model*)
if *P* is non-null **then return** DPLL(*clauses*, *symbols* - *P*, *model* \cup {*P*=*value*})
P, *value* \leftarrow ³ FIND-UNIT-CLAUSE(*clauses*, *model*)
if *P* is non-null **then return** DPLL(*clauses*, *symbols* - *P*, *model* \cup {*P*=*value*})
P \leftarrow FIRST(*symbols*); *rest* \leftarrow REST(*symbols*)
return DPLL(*clauses*, *rest*, *model* \cup {*P*=*true*}) **or**
DPLL(*clauses*, *rest*, *model* \cup {*P*=*false*})

Davis-Putnam procedure



function DP(Δ)

for φ **in** *vocabulary* (Δ) **do**

var $\Delta' \leftarrow \{ \}$;

for Φ_1 **in** Δ **for** Φ_2 **in** Δ such that $\varphi \in \Phi_1 \neg\varphi \in \Phi_2$ **do**

var $\Phi' \leftarrow \Phi_1 - \{\varphi\} \cup \Phi_2 - \{\neg\varphi\}$;

if not *tautology*(Φ') **then** $\Delta' \leftarrow \Delta' \cup \{\Phi'\}$;

$\Delta \leftarrow \Delta - \{\Phi \in \Delta \mid \varphi \in \Phi \text{ or } \neg\varphi \in \Phi\} \cup \Delta'$;

return {**if** $\{ \} \in \Delta$ **then** *unsatisfiable* **else** *satisfiable*};

function *tautology*(Φ)

$\varphi \in \Phi$ and $\neg\varphi \in \Phi$

DPLL procedure vs. DP procedure

- DP can cause a quadratic expansion every time it is applied.
 - This can easily exhaust space on large problems.
- DPLL attacks the problem by sequentially solving smaller problems.
 - Basic idea: Choose a literal. Assume true, simplify clause set, and try to show satisfiable. Repeat for the negation of the literal.
 - Good because we do not cross multiply the clause set

DPLL procedure vs. DP procedure

Problem	Tautology	DP	DPLL
<i>Prime</i>	30.00	0.00	0.00
<i>Prime4</i>	0.02	0.06	0.04
<i>Prime9</i>	18.94	2.98	0.51
<i>Prime10</i>	11.40	3.03	0.96
<i>Prime11</i>	28.11	2.98	0.51
<i>Prime16</i>	> 1 hour	*	9.15
<i>Prime17</i>	> 1 hour	*	3.87
Mkadder32	>> 1 hour	6.50	7.34
Mkadder42	>> 1 hour	22.95	46.86
Mkadder52	>> 1 hour	44.83	170.98
Mkadder53	>> 1 hour	38.27	250.16
Mkadder63	>> 1 hour	*	1186.4
Mkadder73	>> 1 hour	*	3759.9

Reference: <http://logic.stanford.edu/classes/cs157/2011/lectures/lecture04.pdf>



WalkSAT algorithm

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

function WALKSAT(*clauses*, *p*, *max_flips*) **returns** a satisfying model or *failure*

inputs: *clauses*, a set of clauses in propositional logic

p, the probability of choosing to do a “random walk” move, typically around 0.5

max_flips, number of flips allowed before giving up

model \leftarrow a random assignment of *true/false* to the symbols in *clauses*

for *i* = 1 **to** *max_flips* **do**

if *model* satisfies *clauses* **then return** *model*

clause \leftarrow a randomly selected clause from *clauses* that is false in *model*

with probability *p* flip the value in *model* of a randomly selected symbol from *clause*

else flip whichever symbol in *clause* maximizes the number of satisfied clauses

return *failure*

WalkSAT algorithm



- The algorithm returns a model → **satisfiable**
- The algorithm returns false → **unsatisfiable** **OR** more time is needed for searching
- WalkSAT cannot always detect **unsatisfiability**
 - It is most useful **when a solution is expected to exist.**
- For example,
 - An agent cannot *reliably* use WALKSAT to prove that a square is safe in the Wumpus world.
 - Instead, it can say, “I thought about it for an hour and couldn’t come up with a possible world in which the square *isn’t* safe.”

Quiz 06: DPLL and DP

- Given a KB as shown aside

KB

$$A \Rightarrow B \vee C$$

$$A \Rightarrow D$$

$$C \wedge D \Rightarrow \neg F$$

$$B \Rightarrow F$$

$$A$$

- Using either DPLL or DP to check whether KB entails each of the following sentences
 - C
 - $B \Rightarrow \neg C$

...the end.

