

Propositional Logic

Bang Liu, Jian-Yun Nie

IFT3335: Introduction to Artificial Intelligence

Readings: AIMA 7.1~7.5



Outline

- 1 Logical Agents
- 2 Propositional Logic
- 3 Inference
- 4 Resolution and CNF
- 5 Forward Chaining
- 6 Backward Chaining
















Generic Knowledge-Based Agent

- **Knowledge base** is a set of **sentences** in a **formal language**.
- **Declarative** approach to develop an agent: **Tell** it what it needs to know.

KB-AGENT(*percept*)

```
1  TELL(KB, MAKE-PERCEPT-SENTENCE(percept, t))
2  action = ASK(KB, MAKE-ACTION-QUERY(t))
3  TELL(KB, MAKE-ACTION-SENTENCE(action, t))
4  t = t + 1
5  return action
```

Wumpus World

4	 stench		 breeze	 pit
3		 breeze  stench  gold	 pit	 breeze
2	 stench		 breeze	
1		 breeze	 pit	 breeze
	1	2	3	4


Wumpus World (PEAS)

- Performance measure:** gold +1000, death -1000, -1 per move, and -10 for using the arrow.
- Environment:** 4×4 grid. Agent starts at $[1, 1]$, facing right. One gold and one wumpus are uniformly randomly located. Any square can be a pit with probability of 0.2.
- Actuators:** FORWARD, TURNLEFT, TURNRIGHT, GRAB, SHOOT (only one arrow, going straight until it kills the wumpus or hits the wall), CLIMB (only works at $[1, 1]$).
- Sensors:** STENCH when adjacent to the wumpus. BREEZE when adjacent to a pit. GLITTER when reaching the gold. BUMP when walking into a wall. SCREAM when the wumpus dies.




Wumpus World

- The agent sometimes needs to decide to go home empty-handed or risk for the gold.
 - This environment does not guarantee the agent can always get the gold.
 - If at $[1, 1]$ the agent receives BREEZE, the agent does not know which direction to FORWARD is fatal and which is safe (can be both fatal).
 - With a probability of about 21%, the gold is in a pit, or surrounded by pits.
- The agent's initial *KB* contains the rules.
- It also knows it's in $[1, 1]$, and it's a safe square (marked OK).

Wumpus World

4				
3				
2	SAFE			
1		SAFE		
	1	2	3	4

(a)

4				
3				
2	SAFE			
1	SAFE			
	1	2	3	4

(b)




1st step

- Percept: NONE.
- → [1, 2] and [2, 1] are OK.
- Action: FORWARD.








2nd step

- Percept: BREEZE.
- → [2, 2] or [3, 1] are pits.
- → go back to [1, 1] then [1, 2].
- Action: TURNLEFT, TURNLEFT, FORWARD, TURNRIGHT, FORWARD.

Wumpus World

4				
3				
2	SAFE 	SAFE		
1				
	1	2	3	4

(a)

4				
3				
2	SAFE 	SAFE		
1				
	1	2	3	4

(b)

3rd step

- Percept: STENCH.
- → [2, 2]: OK; [1, 3]: wumpus.
- → Kill wumpus; go to [2, 2].
- Action: SHOOT, TURNRIGHT, FORWARD.

5th step

- Percept: STENCH, GLITTER, BREEZE.
- → [2, 4] or [3, 3]: pits; [2, 3]: gold.
- → Get gold and go back.
- Action: GRAB, . . .

Logics

A formal language consists of words whose letters are taken from an alphabet and are well-formed according to a specific set of rules. A formal language is often defined by means of a formal grammar, which consists of its formation rules. Formal languages are languages that are designed by people for specific applications. Tend to have strict syntax and not ambiguous.

- **Logics** are formal languages.
- **Syntax** defines the sentence structures in the language.
- **Semantics** defines the meanings of sentences.
 - Semantics defines the **truth** of each sentence w.r.t. each **possible world**.
 - $x + y = 4$ is true in a world where $x = 1$ and $y = 3$.
- We use the term **model** in place of **possible world**.

Models

If a sentence α is true in model m ,

- m satisfies α .
- m is a model of α .
- $m \in M(\alpha)$, where $M(\alpha)$ is the set of all models of α .

Entailment and Models

Entailment

Knowledge base KB **entails** sentence α **if and only if** α is true in all worlds where KB is true, denoted as:

$$KB \models \alpha$$




- $(x + y = 4) \models (4 = x + y)$.
- $x = 0 \models xy = 0$.

Theorem: $\alpha \models \beta$ iff $M(\alpha) \subseteq M(\beta)$.

only if: $\forall m \in M(\alpha), \beta$ is true in $m \Leftrightarrow \forall m \in M(\alpha), m \in M(\beta) \Leftrightarrow M(\alpha) \subseteq M(\beta)$

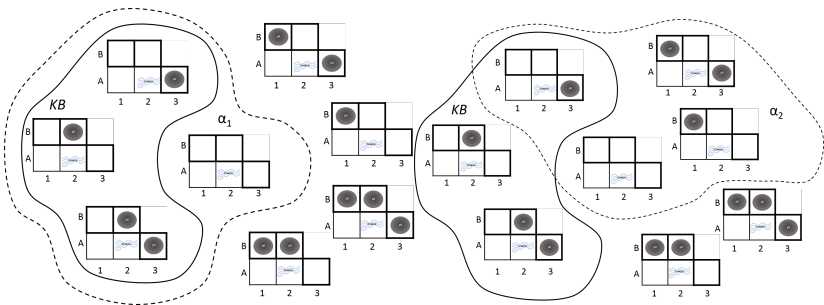
if: $\forall m \in M(\alpha), m \in M(\beta) \Leftrightarrow \forall m$ where α is true, β is true $\Leftrightarrow \alpha \models \beta$. □

Back to Wumpus World

4				
3				
2	SAFE			
1	SAFE			
	1	2	3	4

- $[1, 1]$ percepts NONE.
- $[2, 1]$ percepts BREEZE.
- The agent wants to know unexplored adjacent squares $[1, 2]$, $[2, 2]$, $[3, 1]$ contains pits or not.

- The agent wants to know unexplored adjacent squares $[1, 2]$, $[2, 2]$, $[3, 1]$ contains pits or not.
- $2^3 = 8$ possible models.
- Consider two sentences: α_1 : no pit in $[1, 2]$; α_2 : no pit in $[2, 2]$.



- $KB \models \alpha_1$; $KB \not\models \alpha_2$.

Propositional Logic

- **Proposition**: a declarative sentence that is either **true** or **false**.
 - $\mathcal{P} = \mathcal{NP}$ is a proposition.
 - “How are you?” is not.
- Propositional logic usually does not consider time.
- If the truth of a proposition varies over time, we call it **fluent**.
 - “Today is Monday” is a fluent.
- **Atomic propositions** are minimum propositions.
- **Literals** are atomic propositions or their negations (p or $\neg p$).

Propositional Logic

- The following grammar in **Backus-Naur form (BNF)** for **syntax**.
- **Truth tables** for **semantics**.

<i>Sentence</i>	→	<i>AtomicSentence</i> <i>ComplexSentence</i>
<i>AtomicSentence</i>	→	<i>True</i> <i>False</i> <i>P</i> <i>Q</i> <i>R</i> ...
<i>ComplexSentence</i>	→	(<i>Sentence</i>) () can change operator precedence
		¬ <i>Sentence</i>
		<i>Sentence</i> ∧ <i>Sentence</i>
		<i>Sentence</i> ∨ <i>Sentence</i>
		<i>Sentence</i> ⇒ <i>Sentence</i>
		<i>Sentence</i> ⇔ <i>Sentence</i>

<i>P</i>	<i>Q</i>	¬ <i>P</i>	<i>P</i> ∧ <i>Q</i>	<i>P</i> ∨ <i>Q</i>	<i>P</i> ⇒ <i>Q</i>	<i>P</i> ⇔ <i>Q</i>
<i>false</i>	<i>false</i>	<i>true</i>	<i>false</i>	<i>false</i>	<i>true</i>	<i>true</i>
<i>false</i>	<i>true</i>	<i>true</i>	<i>false</i>	<i>true</i>	<i>true</i>	<i>false</i>
<i>true</i>	<i>false</i>	<i>false</i>	<i>false</i>	<i>true</i>	<i>false</i>	<i>false</i>
<i>true</i>	<i>true</i>	<i>false</i>	<i>true</i>	<i>true</i>	<i>true</i>	<i>true</i>

Figure 7.8 Truth tables for the five logical connectives. To use the table to compute, for example, the value of $P \vee Q$ when P is true and Q is false, first look on the left for the row where P is true and Q is false (the third row). Then look in that row under the $P \vee Q$ column to see the result: *true*.

OPERATOR PRECEDENCE : ¬, ∧, ∨, ⇒, ⇔

not, and, or, implies, if and only if

Inference by Enumeration

TT-ENTAILS(KB, α)

- 1 *symbols* = a list of the proposition symbols in KB and α
- 2 **return** TT-CHECK-ALL($KB, \alpha, symbols, \{\}$)

TT-CHECK-ALL($KB, \alpha, symbols, model$)

- 1 **if** EMPTY(*symbols*)
- 2 **if** PL-TRUE($KB, model$)
- 3 **return** PL-TRUE($\alpha, model$) PL-TRUE returns true if
a sentence holds within a model
- 4 **else return** TRUE
- 5 **else**
- 6 $P =$ FIRST(*symbols*) Check different models/worlds with different
values assigned to symbols
- 7 $rest =$ REST(*symbols*)
- 8 **return** TT-CHECK-ALL($KB, \alpha, rest, model \cup \{P = \text{TRUE}\}$)
 and TT-CHECK-ALL($KB, \alpha, rest, model \cup \{P = \text{FALSE}\}$)

Standard Logical Equivalences

- $\alpha \equiv \beta$ iff $\alpha \models \beta$ and $\beta \models \alpha$.

Note that \equiv is used to make claims about sentences, while \Leftrightarrow is used as a part of a sentence

$$(\alpha \wedge \beta) \equiv (\beta \wedge \alpha)$$

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 and Satisfiability

- A sentence is **valid** if it is true in **all** models.

TRUE, $A \vee \neg A$

- Validity is connected to inference via the **Deduction Theorem**:

$KB \models \alpha$ **iff** $(KB \Rightarrow \alpha)$ is valid.

- A sentence is **satisfiable** if it is true in **some** model.

$A \wedge B, A$

- A sentence is **unsatisfiable** if it is true in **no** model.

$A \wedge \neg A$

- Satisfiability is connected to inference via **Reductio ad Absurdum** (proof by contradiction):

$KB \models \alpha$ **iff** $(KB \wedge \neg \alpha)$ is unsatisfiable.

Inference

Inference: the way we derive conclusions with inference rules.

- Inference i can derive α from KB , denoted as

$$KB \vdash_i \alpha$$

- **Soundness:** i is sound if

$$(KB \vdash_i \alpha) \Rightarrow (KB \models \alpha) \quad \text{i.e., any conclusion I can drive from KB is true}$$




- **Completeness:** i is complete if

$$(KB \models \alpha) \Rightarrow (KB \vdash_i \alpha) \quad \text{i.e., If any sentence is entailed by KB, I can derive it.}$$

- For KB consisting of only propositional logic or first-order logic (FOL), there exists a **sound** and **complete** inference procedure.
- FOL is expressive enough to express many things in the real world.

Simple Knowledge Base Using Propositional Logic

- $P_{x,y}$ is true there is a pit in $[x, y]$.
- $W_{x,y}$ is true there is a wumpus in $[x, y]$.
- $B_{x,y}$ is true if the agent perceives BREEZE in $[x, y]$.
- $S_{x,y}$ is true if the agent perceives STENCH in $[x, y]$.

4				
3				
2	SAFE			
1	SAFE			
	1	2	3	4

KB

- $R_1 : \neg P_{1,1}$.
- $R_2 : B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$.
- $R_3 : B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$.
- $R_4 : \neg B_{1,1}$.
- $R_5 : B_{2,1}$.

Inference of the Wumpus World

Biconditional elimination from 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 from R_6 .

- $R_7 : (P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1}$.

Contrapositive from R_7 .

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

Modus Ponens from R_8 .

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

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

De Morgan's rule from R_9 .

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

Proof by Resolution

- Convert a sentence into **conjunctive normal form (CNF)**.
 - Conjunction of disjunctions of literals clauses.
 - $(A \vee \neg B) \wedge (B \vee \neg C \vee D)$

Resolution Inference Rule

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

where l_i and m_j are complementary literals.

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

Quiz:

If you have $A \vee B \vee \neg C$ and $A \vee \neg B \vee C$, can you derive A ?

Conjunctive Normal Form (CNF) Conversion

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

- ① Eliminate \Leftrightarrow by bidirectional elimination:

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

- ② Eliminate \Rightarrow by $(\alpha \Rightarrow \beta) \equiv (\neg\alpha \vee \beta)$:

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

- ③ “Move \neg inwards” by double-negation elimination and De Morgan:

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

- ④ Distribute \vee over \wedge and “flatten”:

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

Resolution Algorithm

- Proof by contradiction — showing $KB \wedge \neg\alpha$ is unsatisfiable.

PL-RESOLUTION(KB, α)

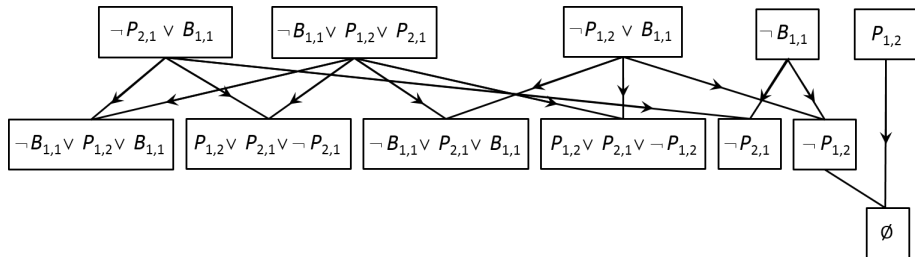
```

1  clauses = the set of clauses in the CNF representation of  $KB \wedge \neg\alpha$ 
2  new =  $\phi$ 
3  repeat
4      for each pair of clauses  $C_i, C_j$  in clauses do
5          resolvents = PL-RESOLVE( $C_i, C_j$ )
6          if resolvents contains the empty clause     $A \wedge \neg A$  leads to empty clause
7              return TRUE
8          new = new  $\cup$  resolvents
9      if new  $\subseteq$  clauses
10         return FALSE
11     clauses = clauses  $\cup$  new

```

Resolution Example

- $KB : B_{1,1} \Leftrightarrow P_{1,2} \vee P_{2,1}$.
- $KB(CNF) : (\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge (\neg P_{1,2} \vee B_{1,1}) \wedge (\neg P_{2,1} \vee B_{1,1})$.
- After we know (add into KB) $\neg B_{1,1}$, we'd like to assert $\alpha = \neg P_{1,2}$.
- PL-RESOLUTION resolves $KB \wedge \neg\alpha$ to the empty clause.



Resolution is Sound and Complete

- Soundness is not surprising since inference rules are sound (check the truth table).
- Resolution is also complete.
 - Resolution closure $RC(S)$ of a set of clauses S : the set of all clauses derivable by resolution.
 - Final value of *clauses* in PL-RESOLUTION is $RC(S)$.
 - $RC(S)$ is finite, and hence PL-RESOLUTION always terminates.

Ground Resolution Theorem

S is unsatisfiable $\Rightarrow RC(S)$ contains the empty clause.

Ground Resolution Theorem

Proof by Contrapositive.

$RC(S)$ does not contain the empty set $\Rightarrow S$ is satisfiable.

If $\phi \notin RC(S)$, construct a model for S with suitable values for literals P_1, \dots, P_k :

For i from 1 to k ,

- If a clause in $RC(S)$ contains $\neg P_i$ and all its other literals are FALSE, assign FALSE to P_i .
- Otherwise, assign TRUE to P_i .

For a clause in S to be close to FALSE, it must be either $(\text{FALSE} \vee \dots \vee \text{FALSE} \vee P_i)$ or $(\text{FALSE} \vee \dots \vee \text{FALSE} \vee \neg P_i)$.

However, our assignment will make the clause to be true. Therefore, such assignment is a model of S . □

Horn and Definite Clauses

- The completeness of resolution is good.
- For many real-world applications, if we add some restrictions, more efficient inference can be achieved.
- **Definite clause**: a disjunction of literals where **exactly one** is positive.
- **Horn clause**: a disjunction of literals where **at most one** is positive.
- Horn clauses are **closed** under resolution:

Resolving two Horn clauses yields a Horn clause.

- Another way to view Horn clauses:
 - $\text{TRUE} \Rightarrow \text{symbol}$.
 - $(\text{Conjunction of symbols}) \Rightarrow \text{symbol}$.

Horn clauses means entailment:

$$\begin{aligned} & \neg A \vee \neg B \vee C \\ \equiv & \neg(A \wedge B) \vee C \\ \equiv & (A \wedge B) \Rightarrow C \end{aligned}$$

- Deciding entailment with Horn clauses can be done in **linear** time!
Forward and backward chaining.

Forward Chaining

- Resolution for Horn clauses (Modus Ponens):

$$\frac{\alpha_1, \dots, \alpha_n, (\alpha_1 \wedge \dots \wedge \alpha_n) \Rightarrow \beta}{\beta}$$

- Main idea:
 - Counts the unknown premises in all clauses.
 - Decreases the count if a premise is known.
 - When a count becomes zero, the conclusion is added as a known fact.
 - Record the inferred symbols to avoid redundant work
($P \Rightarrow Q, Q \Rightarrow P$).

Forward Chaining

PL-FC-ENTAILS(KB, q)

count: number of symbols in c 's premise.

agenda: a queue of symbols, initially known facts in KB.

```
1  while agenda is not empty do
2       $p = \text{POP}(\textit{agenda})$ 
3      if  $p == q$ 
4          return TRUE
5      if  $\textit{inferred}[p] == \text{FALSE}$ 
6           $\textit{inferred}[p] = \text{TRUE}$ 
7          for each clause  $c$  in  $KB$  where  $p$  is in  $c.\textit{premise}$ 
8              decrement  $\textit{count}[c]$ 
9              if  $\textit{count}[c] == 0$ 
10                 add  $c.\textit{conclusion}$  to agenda
11 return FALSE
```

Forward Chaining Example

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

KB

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

Step 1

$P \Rightarrow Q$ 1
 $L \wedge M \Rightarrow P$ 2
 $B \wedge L \Rightarrow M$ 2
 $A \wedge P \Rightarrow L$ 2
 $A \wedge B \Rightarrow L$ 2
agenda : [A, B]

Step 2

$P \Rightarrow Q$ 1
 $L \wedge M \Rightarrow P$ 2
 $B \wedge L \Rightarrow M$ 2
 $A \wedge P \Rightarrow L$ 1
 $A \wedge B \Rightarrow L$ 1
agenda : [B]

Step 3

$P \Rightarrow Q$ 1
 $L \wedge M \Rightarrow P$ 2
 $B \wedge L \Rightarrow M$ 1
 $A \wedge P \Rightarrow L$ 1
 $A \wedge B \Rightarrow L$ 0
agenda : [L]

Forward Chaining Example

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

Step 4

$P \Rightarrow Q$	1
$L \wedge M \Rightarrow P$	1
$B \wedge L \Rightarrow M$	0
$A \wedge P \Rightarrow L$	1
$A \wedge B \Rightarrow L$	0
agenda : [M]	

Step 5

$P \Rightarrow Q$	1
$L \wedge M \Rightarrow P$	0
$B \wedge L \Rightarrow M$	0
$A \wedge P \Rightarrow L$	1
$A \wedge B \Rightarrow L$	0
agenda : [P]	

Step 6

$P \Rightarrow Q$	0
$L \wedge M \Rightarrow P$	0
$B \wedge L \Rightarrow M$	0
$A \wedge P \Rightarrow L$	0
$A \wedge B \Rightarrow L$	0
agenda : [Q, L]	

Completeness of Forward Chaining

- FC reaches a **fixed point** where no new inferences are possible.
- Consider a model m which assigns TRUE to every symbol **inferred** and FALSE to others.
- Every clause in KB is TRUE in m :

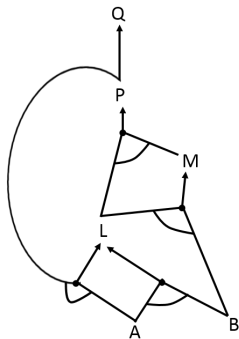
Proof.

- Suppose $\alpha_1 \wedge \dots \wedge \alpha_k \Rightarrow \beta$ is FALSE in m .
- $\alpha_1 \wedge \dots \wedge \alpha_k$ is TRUE and β is FALSE.
- FC has not reached a fixed point. □

- $m \in M(KB)$ (m is a model of KB)
- If $KB \models q$, q is TRUE in m .
- Therefore, FC derives **every** atomic sentence that is **entailed** by KB .

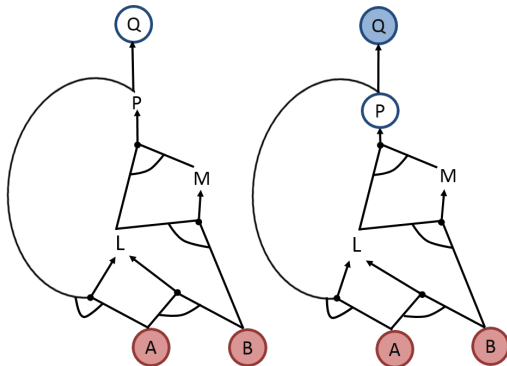
Backward Chaining

- Work backward from Q .
- If Q is known, done.
- Otherwise, check its premise P .
- Next step checks L and M .
- Identical to the AND-OR-GRAPH-SEARCH in the textbook.

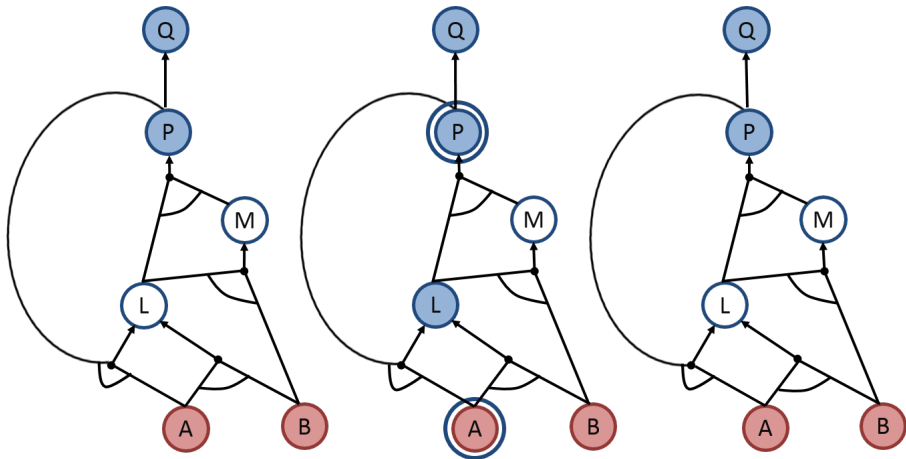


Backward Chaining Example

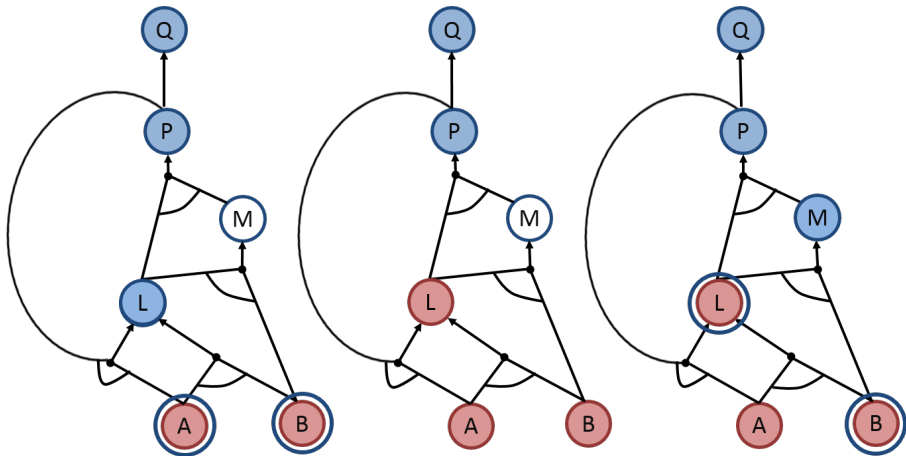
Blue ring: Checking.
Blue circle: Checked.
Red circle: Facts.



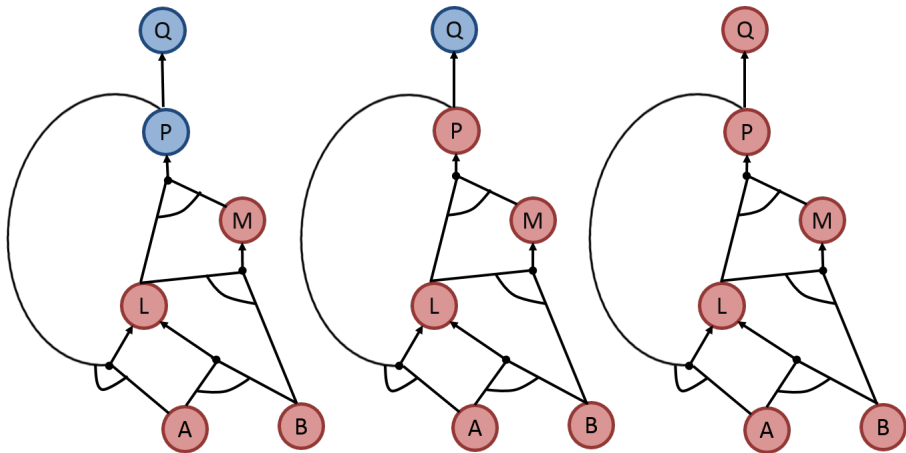
Backward Chaining Example



Backward Chaining Example



Backward Chaining Example



Forward vs. Backward Chaining

- Both time complexities are **linear** to the size of *KB*.
- Forward chaining is **data-driven**.
- Backward chaining is **goal-driven**.
- In general, backward chaining is more appropriate for problem solving
 - Where's my key?
 - How to pass this course?
- Forward chaining may generate many conclusions irrelevant to the goal.
- In general, time complexity of backward chaining is **much less** than linear of the size of *KB*.

Pros and Cons of Propositional Logic

- Pro

- Propositional logic is **declarative**: pieces of syntax correspond to facts.
- Propositional logic allows partial/disjunctive/negated information (unlike most data structures and databases).
- Propositional logic is **compositional**:
 - Meaning of $B_{1,1} \wedge P_{1,2}$ is derived from meaning of $B_{1,1}$ and of $P_{1,2}$.
- Meaning in propositional logic is **context-independent** (unlike natural language, where meaning depends on context)

- Con

- Propositional logic has very limited expressive power.
 - Cannot say “pits cause breezes in adjacent squares” except by writing one sentence for each square.

Summary

- **Knowledge base** contains **sentences** in a knowledge representation language.
- A representation language is defined by its **syntax** and **semantics**, which defines the **truth** of each sentence in each **model**.
- α **entails** β if β is true in all models where α is true. Equivalent definitions: **validity** of $\alpha \Rightarrow \beta$; **unsatisfiability** of $\alpha \wedge \neg\beta$.
- **Sound** inferences derive **only** sentences that are entailed; **complete** inferences derive **all** sentences that are entailed.
- **Resolution** is sound and complete inference for propositional logics, where KB can be expressed by **CNF**.
- **Forward and backward chaining** are sound and complete for KB in **Horn form** (more restrict than propositional logics).