# Propositional Logic

Tian-Li Yu

Taiwan Evolutionary Intelligence Laboratory (TEIL)
Department of Electrical Engineering
National Taiwan University
tianliyu@ntu.edu.tw

Readings: AIMA 7.1∼7.5

# Outline

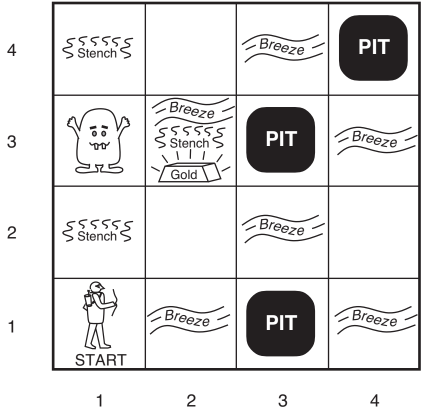
# Generic Knowledge-Based Agent

- 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*)**

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

# Wumpus World



# Wumpus World (PEAS)

**Performance measure:** gold $+1000$, death -1000, -1 per move, and -10 for using the arrow.

**Environment:** $4 \times 4$ 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.

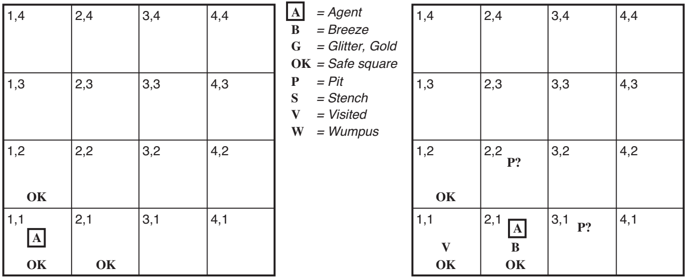
**Actuators:** Forward, TurnLeft, TurnRight, Grab, Shoot (only one arrow, going straight until it kills the wumpus or hits the wall), Climb (only works at $[1, 1]$).

**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



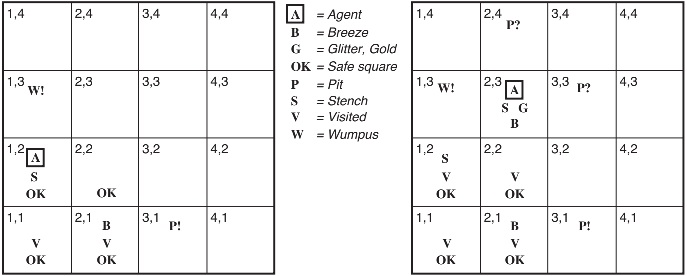

## 1$^{st}$ step

- Percept: NONE.
- $\rightarrow$ $[1, 2]$ and $[2, 1]$ are OK.
- Action: FORWARD.

## 2$^{nd}$ step

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

# Wumpus World




## $3^{rd}$ step

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

## $5^{th}$ step

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

# Logics

- 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 $\alpha$ is true in model $m$,

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

# Entailment and Models

**Entailment**

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

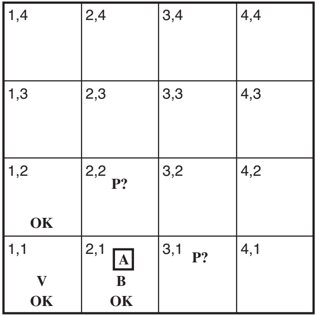
$$KB \models \alpha$$

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

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

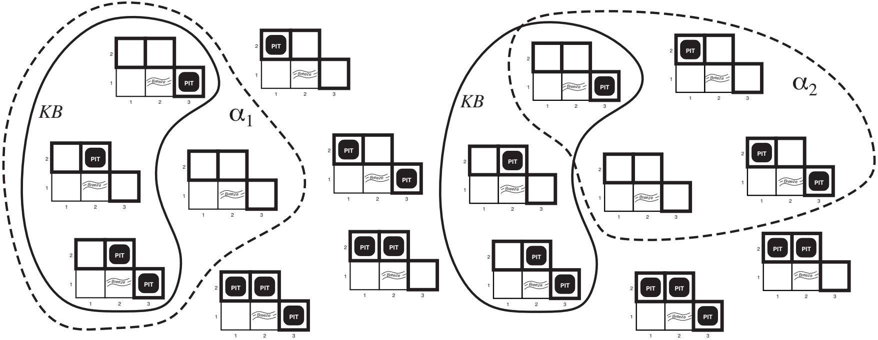
**only if:** $\forall m \in M(\alpha)$, $\beta$ is true in $m \Leftrightarrow \forall m \in M(\alpha), m \in M(\beta) \Leftrightarrow M(\alpha) \subseteq M(\beta)$
**if:** $\forall m \in M(\alpha), m \in M(\beta) \Leftrightarrow \forall m$ where $\alpha$ is true, $\beta$ is true $\Leftrightarrow \alpha \models \beta$. $\square$

# Back to Wumpus World




- [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: $\alpha_1$: no pit in $[1,2]$; $\alpha_2$: 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.

$$
\begin{array}{rcl}
\mathit{Sentence} & \rightarrow & \mathit{AtomicSentence} \mid \mathit{ComplexSentence} \\
\mathit{AtomicSentence} & \rightarrow & \mathit{True} \mid \mathit{False} \mid P \mid Q \mid R \mid \ldots \\
\mathit{ComplexSentence} & \rightarrow & (\mathit{Sentence}) \mid [\mathit{Sentence}] \\
 & \mid & \neg \mathit{Sentence} \\
 & \mid & \mathit{Sentence} \land \mathit{Sentence} \\
 & \mid & \mathit{Sentence} \lor \mathit{Sentence} \\
 & \mid & \mathit{Sentence} \Rightarrow \mathit{Sentence} \\
 & \mid & \mathit{Sentence} \Leftrightarrow \mathit{Sentence} \\
\textsc{Operator Precedence} & : & \neg, \land, \lor, \Rightarrow, \Leftrightarrow
\end{array}
$$

# Inference by Enumeration

### TT-ENTAILS($KB, \alpha$)

```
symbols = a list of the proposition symbols in KB and α
return TT-CHECK-ALL(KB, α, symbols, {})
```

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

```
if EMPTY(symbols)
    if PL-TRUE(KB, model)
        return PL-TRUE(α, model)
    else return TRUE
else
    P = FIRST(symbols)
    rest = REST(symbols)
    return TT-CHECK-ALL(KB, α, rest, model ∪ {P = TRUE})
        and TT-CHECK-ALL(KB, α, rest, model ∪ {P = FALSE})
```

# Standard Logical Equivalences

- $\alpha \equiv \beta$ **iff** $\alpha \models \beta$ 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 and Satisfiability

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

  $\textsc{True}$, $A \vee \neg A$

- 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 $i$ can derive $\alpha$ from $KB$, denoted as

  $$KB \vdash_i \alpha$$

- Soundness: $i$ is sound if

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

- Completeness: $i$ is complete if

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

- 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 $[x, y]$.

| | | | |
|---|---|---|---|
| 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 $\boxed{\text{A}}$ B OK | 3,1 P? | 4,1 |

**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 $R_8$.

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

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 $\underbrace{\text{disjunctions of literals}}_{\text{clauses}}$.
  - $(A \vee \neg B) \wedge (B \vee \neg C \vee D)$

## Resolution Inference Rule

$$\frac{\ell_1 \vee \cdots \vee \ell_k, \quad m_1 \vee \cdots \vee m_n}{\ell_1 \vee \cdots \ell_{i-1} \vee \ell_{i+1} \vee \cdots \vee \ell_k \vee m_1 \vee \cdots m_{j-1} \vee m_{j+1} \vee \cdots \vee m_n},$$

where $\ell_i$ and $m_j$ are complementary literals.

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

# 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

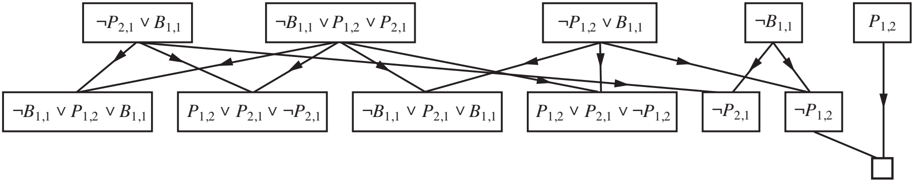
- Proof by contradiction — showing $KB \wedge \neg\alpha$ is unsatisfiable.

**PL-RESOLUTION($KB, \alpha$)**

```
clauses = the set of clauses in the CNF representation of KB ∧ ¬α
new = ϕ
repeat
    for each pair of clauses C_i, C_j in clauses do
        resolvents = PL-RESOLVE(C_i, C_j)
        if resolvents contains the empty clause
            return TRUE
        new = new ∪ resolvents
    if new ⊆ clauses
        return FALSE
    clauses = clauses ∪ 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 contains the empty set $\Rightarrow$ $S$ is satisfiable.

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

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 $(\text{FALSE} \vee \cdots \text{FALSE} \vee P_i)$ or $(\text{FALSE} \vee \cdots \text{FALSE} \vee \neg P_i)$.

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

# 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:
  - $\text{TRUE} \Rightarrow$ symbol.
  - (Conjunction of symbols) $\Rightarrow$ symbol.
- Deciding entailment with Horn clauses can be done in linear time!

  Forward and backward chaining.

# Forward Chaining

- Resolution for Horn clauses (Modus Ponens):

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

- 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.

```
while agenda is not empty do
    p = POP(agenda)
    if p == q
        return TRUE
    if inferred[p] == FALSE
        inferred[p] = TRUE
        for each clause c in KB where p is in c.premise
            decrement count[c]
            if count[c] == 0
                add c.conclusion to agenda
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 $m$:

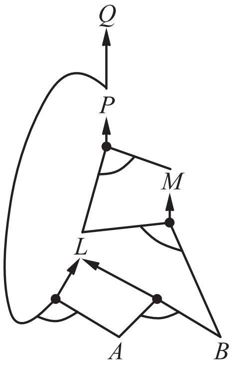
**Proof.**

- Suppose $\alpha_1 \wedge \cdots \wedge \alpha_k \Rightarrow \beta$ is FALSE in $m$.
- $\alpha_1 \wedge \cdots \wedge \alpha_k$ is TRUE and $\beta$ 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 $KB$.

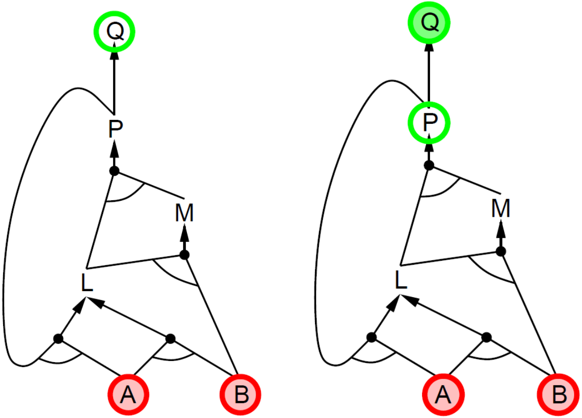
# Backward Chaining

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

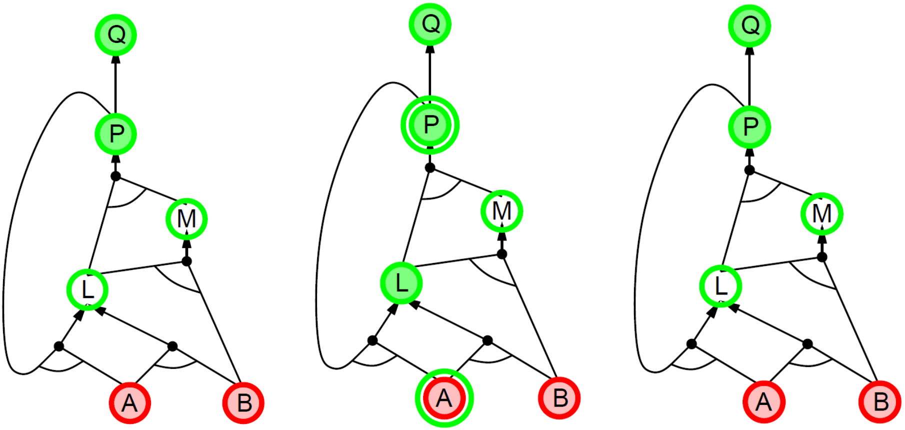


# Backward Chaining Example

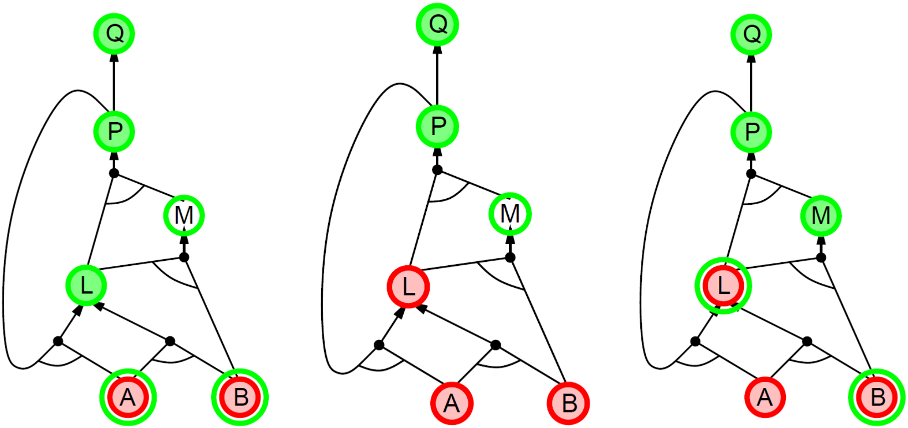
Green ring: Checking.
Green circle: Checked.
Red circle: Facts.



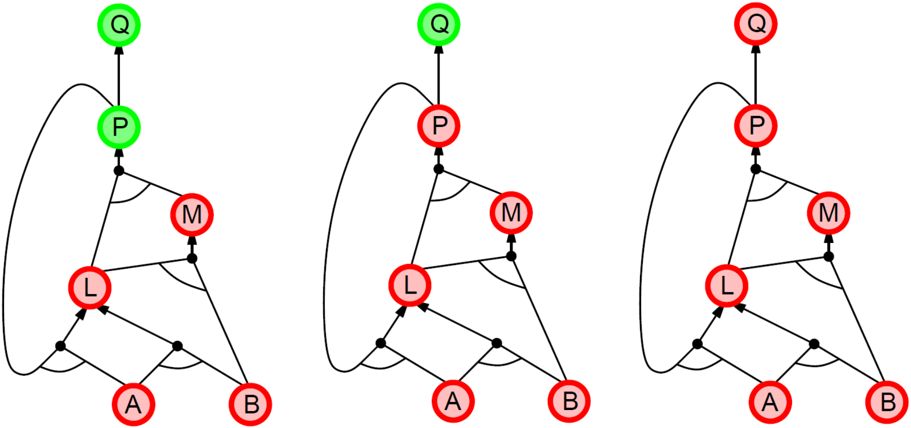
# Backward Chaining Example



# 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.
- $\alpha$ entails $\beta$ if $\beta$ is true in all models where $\alpha$ is true. Equivalent definitions: validity of $\alpha \Rightarrow \beta$; unsatisfiability of $\alpha \land \neg\beta$.
- 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).