

# CS 480

## *Introduction to Artificial Intelligence*

October 11, 2022

# Announcements / Reminders

- **Midterm Exam: THIS THURSDAY (October 13th)!**
  - **Online section:** please make arrangements. Contact Mr. Charles Scott (scott@iit.edu) if in doubt
- **Written Assignment #02:**
  - due: October 16th, 11:00 PM CST
- **Programming Assignment #01:**
  - due: October 15th, 11:00 PM CST
- Please follow the Week 07 To Do List instructions
- Spring Semester midterm course evaluation:
  - **Thank you for your feedback!**
- Grading TA assignment:

[https://docs.google.com/spreadsheets/d/1ExS0bKnGt\\_fdf4LHa3YS1qRA7-lq4xqXVjfSAPMaGVk/edit?usp=sharing](https://docs.google.com/spreadsheets/d/1ExS0bKnGt_fdf4LHa3YS1qRA7-lq4xqXVjfSAPMaGVk/edit?usp=sharing)

# Plan for Today

- **Proof by Resolution**

# Proof by Resolution

The process of proving by resolution is as follows:

- A. Formalize the problem: “English to Propositional Logic”
- B. derive  $KB \wedge \neg Q$  (<- negated KB entails Q)
- C. convert  $KB \wedge \neg Q$  into CNF (“standardized”) form
- D. Apply resolution rule to resulting clauses. New clauses will be generated (add them to the set if not already present)
- E. Repeat (C) until:
  - a. no new clause can be added (KB does NOT entail Q)
  - b. last two clauses resolve to yield the empty clause (KB entails Q)

# Logical Entailment

So far, we have been asking the question:

“Does **KB** entail **Q** (does **Q** follow from **KB**)?”

**KB**  $\square$  **Q**

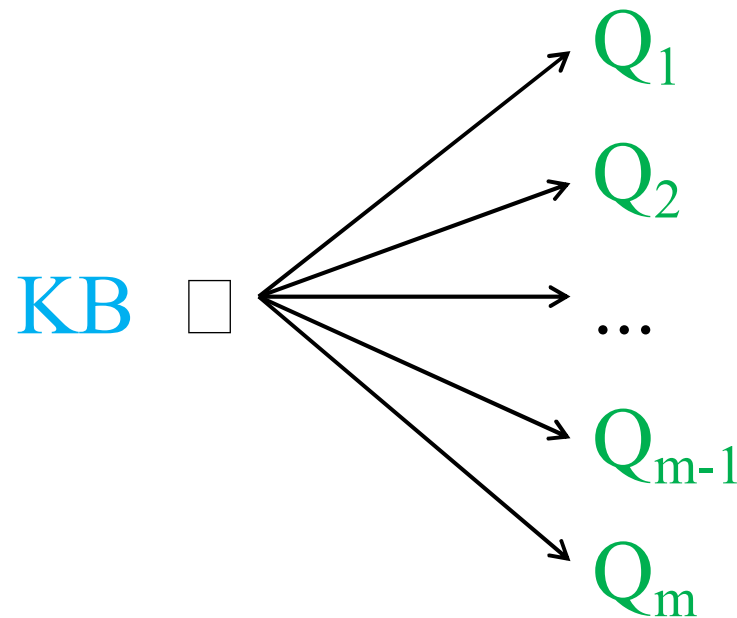
But we could ask the following question:

“Which **Q**s follow from **KB**?”

# Logical Entailment

But we could ask the following question:

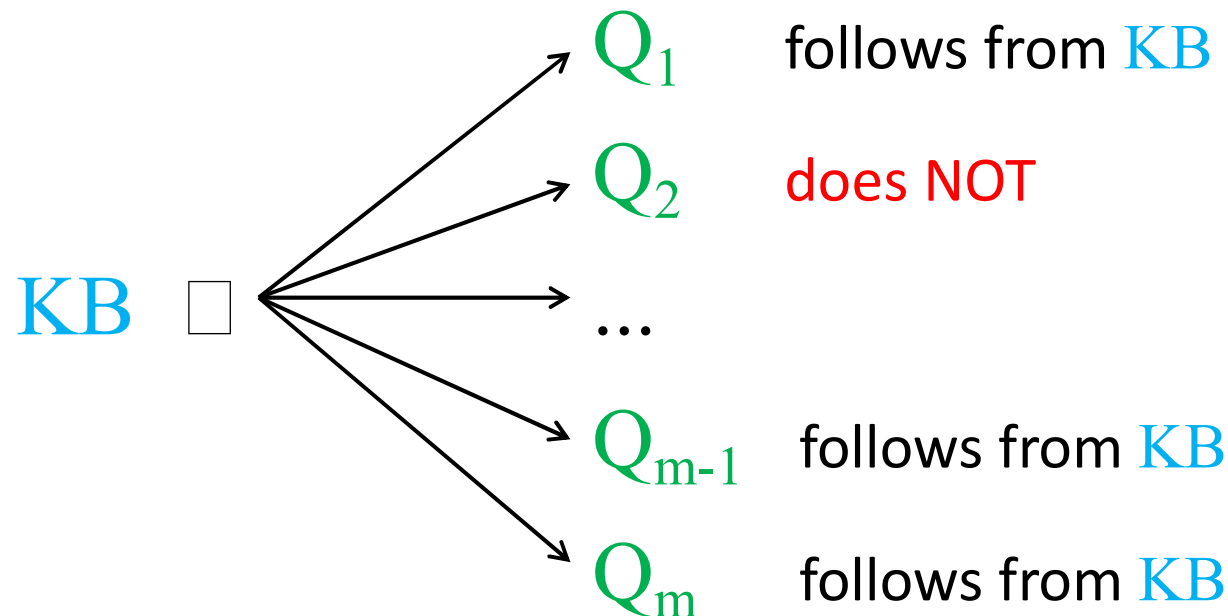
“Which  $Q$ s follow from  $KB$ ?”



# Logical Entailment

But we could ask the following question:

“Which  $Q$ s follow from  $KB$ ?”



# KB Agents

**Sentences are physical configurations of the agent, and reasoning is a process of constructing new physical configurations from old ones.**

**Logical reasoning should ensure that the new configurations represent aspects of the world that actually follow from the aspects that the old configurations represent.**



# Knowledge-based Agents

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

KB BEFORE

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*

CURRENT KB

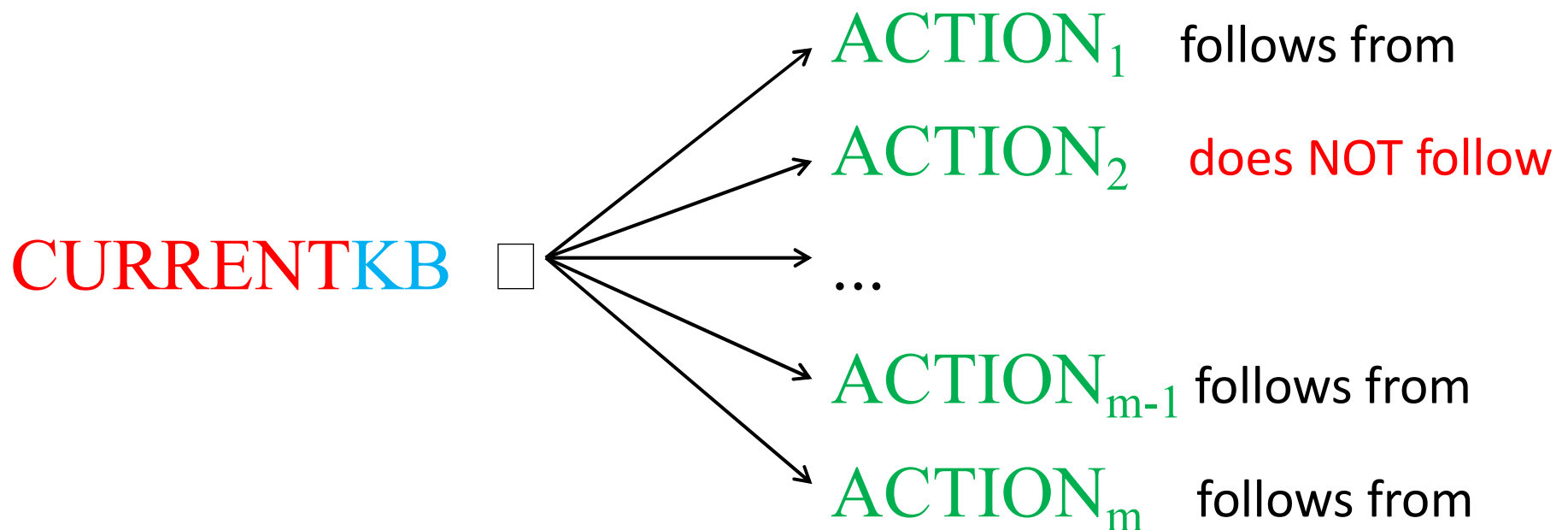
new percept

$\text{CURRENTKB} \Leftrightarrow \text{KB BEFORE} \wedge \text{percept}$

# Logical Entailment with KB Agents

But we could ask the following question:

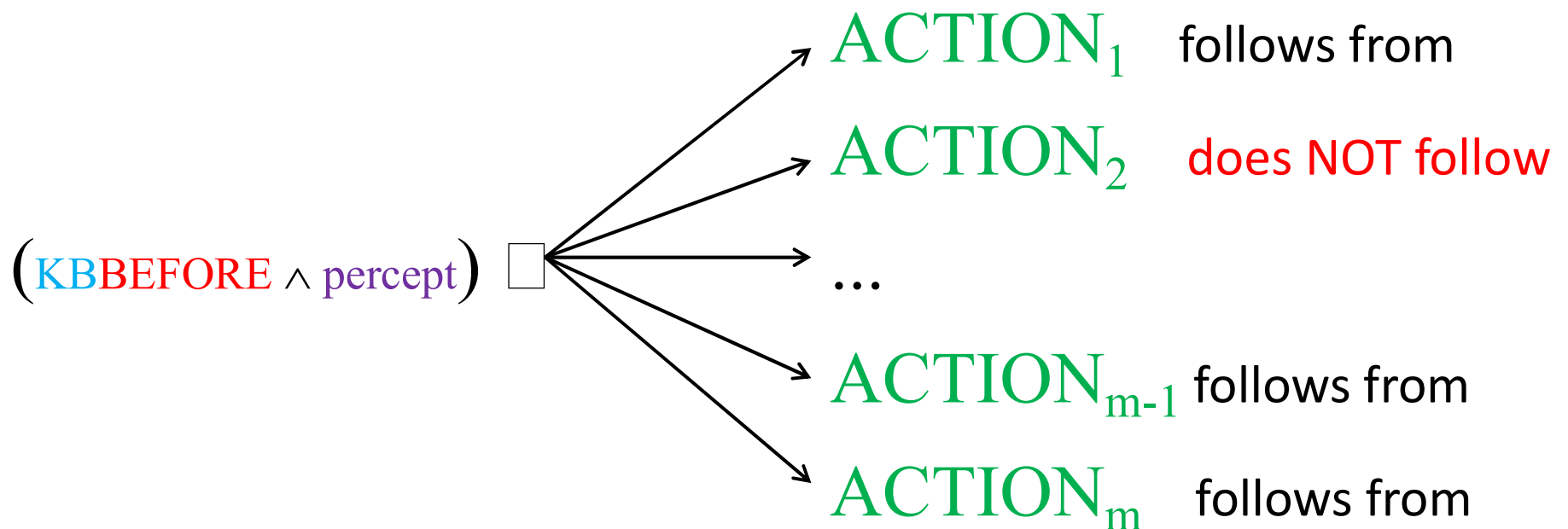
“Which **ACTION**s follow from **CURRENTKB**?”



# Logical Entailment with KB Agents

But we could ask the following question:

“Which **ACTION**s follow from **CURRENTKB**?”



# Logical Entailment with KB Agents

Let's try a simpler example with just ONE ACTION to consider. The question is:

“Does ACTION follow from CURRENTKB?”

Test / prove:

$(\text{KB}_{\text{BEFORE}} \wedge \text{percept}) \sqsubseteq \text{ACTION}$  follows from

to decide whether to apply ACTION or not.

# KB Agent: Should I Stay or Should I Go



**Problem: KB Agent wants to cross the street. Traffic comes from left and right. KB agent cannot cross if there is ANY traffic (ignore the STOP sign).**



# Proof by Resolution

The process of proving by resolution is as follows:

- A. Formalize the problem: “English to Propositional Logic”
- B. derive  $KB \wedge \neg Q$
- C. convert  $KB \wedge \neg Q$  into CNF (“standardized”) form
- D. Apply resolution rule to resulting clauses. New clauses will be generated (add them to the set if not already present)
- E. Repeat (C) until:
  - a. no new clause can be added ( $KB$  does NOT entail  $Q$ )
  - b. last two clauses resolve to yield the empty clause ( $KB$  entails  $Q$ )

# Street Crosser: Convert KB to CNF

Variant A:  $\text{KB} \equiv \text{walk} \Leftrightarrow (\neg \text{trafficLeft} \wedge \neg \text{trafficRight})$

$$(\text{walk} \Rightarrow (\neg \text{trafficLeft} \wedge \neg \text{trafficRight})) \wedge ((\neg \text{trafficLeft} \wedge \neg \text{trafficRight}) \Rightarrow \text{walk})$$

by Biconditional Elimination

$$(\neg \text{walk} \vee (\neg \text{trafficLeft} \wedge \neg \text{trafficRight})) \wedge (\neg(\neg \text{trafficLeft} \wedge \neg \text{trafficRight}) \vee \text{walk})$$

by Implication Elimination

$$((\neg \text{walk} \vee \neg \text{trafficLeft}) \wedge (\neg \text{walk} \vee \neg \text{trafficRight})) \wedge (\neg(\neg \text{trafficLeft} \wedge \neg \text{trafficRight}) \vee \text{walk})$$

by Distributivity Rule

$$((\neg \text{walk} \vee \neg \text{trafficLeft}) \wedge (\neg \text{walk} \vee \neg \text{trafficRight})) \wedge ((\neg \neg \text{trafficLeft} \vee \neg \neg \text{trafficRight}) \vee \text{walk})$$

by De Morgan's Rule

$$((\neg \text{walk} \vee \neg \text{trafficLeft}) \wedge (\neg \text{walk} \vee \neg \text{trafficRight})) \wedge ((\text{trafficLeft} \vee \text{trafficRight}) \vee \text{walk})$$

by Double Negation Elimination

$$(\neg \text{walk} \vee \neg \text{trafficLeft}) \wedge (\neg \text{walk} \vee \neg \text{trafficRight}) \wedge (\text{trafficLeft} \vee \text{trafficRight} \vee \text{walk})$$

remove extraneous parentheses

# Should I **Go**: Proof by Resolution

We have our knowledge base KB in CNF ready:

$$KB \equiv (\neg \text{walk} \vee \neg \text{trafficLeft}) \wedge (\neg \text{walk} \vee \neg \text{trafficRight}) \wedge (\text{trafficLeft} \vee \text{trafficRight} \vee \text{walk})$$

Let's rename propositional variables to simplify:

$$KB \equiv (\text{w} \vee \text{tL} \vee \text{tR}) \wedge (\neg \text{tL} \vee \neg \text{w}) \wedge (\neg \text{tR} \vee \neg \text{w})$$

Assume that traffic is **NOT** coming from both left and right (percepts):

$$\text{PERCEPTS} \equiv (\neg \text{tR}) \wedge (\neg \text{tL})$$

Let's add (TELL) PERCEPTS to the Knowledge Base KB:

$$KB_N \equiv KB \wedge \text{PERCEPTS}$$

$$KB_N \equiv (\text{w} \vee \text{tL} \vee \text{tR}) \wedge (\neg \text{tL} \vee \neg \text{w}) \wedge (\neg \text{tR} \vee \neg \text{w}) \wedge (\neg \text{tR}) \wedge (\neg \text{tL})$$

Our query Q is “**Should I (choose action) walk?**”:

$$Q \equiv \text{w} \text{ (and in negated form: } \neg Q \equiv \neg \text{w)}$$

To test / prove entailment I want to prove that  $KB_N \wedge \neg Q$  is true:

$$KB_N \wedge \neg Q \equiv (\text{w} \vee \text{tL} \vee \text{tR}) \wedge (\neg \text{tL} \vee \neg \text{w}) \wedge (\neg \text{tR} \vee \neg \text{w}) \wedge (\neg \text{tR}) \wedge (\neg \text{tL}) \wedge (\neg \text{w})$$

$$(\text{w} \vee \text{tL} \vee \text{tR})_1 \wedge (\neg \text{tL} \vee \neg \text{w})_2 \wedge (\neg \text{tR} \vee \neg \text{w})_3 \wedge (\neg \text{tR})_4 \wedge (\neg \text{tL})_5 \wedge (\neg \text{w})_6$$



# Should I Go: Proof by Resolution

Prove:

$$\text{KB}_N \wedge \neg Q \equiv$$
$$(\textcolor{blue}{w} \vee \textcolor{green}{tL} \vee \textcolor{red}{tR})_1 \wedge (\neg \textcolor{green}{tL} \vee \neg \textcolor{blue}{w})_2 \wedge (\neg \textcolor{red}{tR} \vee \neg \textcolor{blue}{w})_3 \wedge (\neg \textcolor{red}{tR})_4 \wedge (\neg \textcolor{green}{tL})_5 \wedge (\neg \textcolor{blue}{w})_6$$

# Should I Go: Proof by Resolution

Prove:

$$KB_N \wedge \neg Q \equiv$$

$$(w \vee tL \vee tR)_1 \wedge (\neg tL \vee \neg w)_2 \wedge (\neg tR \vee \neg w)_3 \wedge (\neg tR)_4 \wedge (\neg tL)_5 \wedge (\neg w)_6$$

Known clauses:

1. ( $w \vee tL \vee tR$ )
2. ( $\neg tL \vee \neg w$ )
3. ( $\neg tR \vee \neg w$ )
4. ( $\neg tR$ )
5. ( $\neg tL$ )
6. ( $\neg w$ )

Added clauses:

# Proof by Resolution: Example

Prove:

$$\text{KB}_N \wedge \neg Q \equiv$$
$$(\text{w} \vee \text{tL} \vee \text{tR})_1 \wedge (\neg \text{tL} \vee \neg \text{w})_2 \wedge (\neg \text{tR} \vee \neg \text{w})_3 \wedge (\neg \text{tR})_4 \wedge (\neg \text{tL})_5 \wedge (\neg \text{w})_6$$

Known clauses:

1. ( $\text{w} \vee \text{tL} \vee \text{tR}$ )
2. ( $\neg \text{tL} \vee \neg \text{w}$ )
3. ( $\neg \text{tR} \vee \neg \text{w}$ )
4. ( $\neg \text{tR}$ )
5. ( $\neg \text{tL}$ )
6. ( $\neg \text{w}$ )

Added clauses:

7. ( $\text{tL} \vee \text{tR}$ )

Resolution applied to clauses 1 and 6

$$\frac{(\text{w} \vee \text{tL} \vee \text{tR}), (\neg \text{w})}{(\text{tL} \vee \text{tR})}$$

Produces a new clause ( $\text{tL} \vee \text{tR}$ ). We can add it to the list as clause (7).

# Proof by Resolution: Example

Prove:

$$\text{KB}_N \wedge \neg Q \equiv$$
$$(\text{w} \vee \text{tL} \vee \text{tR})_1 \wedge (\neg \text{tL} \vee \neg \text{w})_2 \wedge (\neg \text{tR} \vee \neg \text{w})_3 \wedge (\neg \text{tR})_4 \wedge (\neg \text{tL})_5 \wedge (\neg \text{w})_6$$

Known clauses:

1.  $(\text{w} \vee \text{tL} \vee \text{tR})$
2.  $(\neg \text{tL} \vee \neg \text{w})$
3.  $(\neg \text{tR} \vee \neg \text{w})$
4.  $(\neg \text{tR})$
5.  $(\neg \text{tL})$
6.  $(\neg \text{w})$

Added clauses:

7.  $(\text{tL} \vee \text{tR})$
8.  $(\text{tL})$

Resolution applied to clauses 4 and 7

$$\frac{(\neg \text{tR}), (\text{tL} \vee \text{tR})}{(\text{tL})}$$

Produces a new clause  $(\text{tL})$ . We can add it to the list as clause (8).

# Proof by Resolution: Example

Prove:

$$\text{KB}_N \wedge \neg Q \equiv \\ (\text{w} \vee \text{tL} \vee \text{tR})_1 \wedge (\neg \text{tL} \vee \neg \text{w})_2 \wedge (\neg \text{tR} \vee \neg \text{w})_3 \wedge (\neg \text{tR})_4 \wedge (\neg \text{tL})_5 \wedge (\neg \text{w})_6$$

Known clauses:

1. ( $\text{w} \vee \text{tL} \vee \text{tR}$ )
2. ( $\neg \text{tL} \vee \neg \text{w}$ )
3. ( $\neg \text{tR} \vee \neg \text{w}$ )
4. ( $\neg \text{tR}$ )
5. ( $\neg \text{tL}$ )
6. ( $\neg \text{w}$ )

Added clauses:

7. ( $\text{tL} \vee \text{tR}$ )
8. ( $\text{tL}$ )

Resolution applied to clauses 5 and 8

$$(\neg \text{tL}), (\text{tL})$$

---

()

Produces an empty clause / contradiction. Stop.

# Proof by Resolution: Example

Prove:

$$KB_N \wedge \neg Q \equiv$$

$$(\textcolor{blue}{w} \vee \textcolor{green}{tL} \vee \textcolor{red}{tR})_1 \wedge (\neg \textcolor{green}{tL} \vee \neg \textcolor{blue}{w})_2 \wedge (\neg \textcolor{red}{tR} \vee \neg \textcolor{blue}{w})_3 \wedge (\neg \textcolor{red}{tR})_4 \wedge (\neg \textcolor{green}{tL})_5 \wedge (\neg \textcolor{blue}{w})_6$$

At this point, we tried to resolve all promising clause pairs and we reached an empty clause  $\rightarrow$  KB **entails** Q.

Given PERCEPTS:  $(\neg \textcolor{red}{tR}) \wedge (\neg \textcolor{green}{tL})$

we should apply action walk (**w**) and **go**.

Known clauses:

1. (**w**  $\vee$  **tL**  $\vee$  **tR**)
2. ( $\neg \textcolor{green}{tL}$   $\vee$   $\neg \textcolor{blue}{w}$ )
3. ( $\neg \textcolor{red}{tR}$   $\vee$   $\neg \textcolor{blue}{w}$ )
4. ( $\neg \textcolor{red}{tR}$ )
5. ( $\neg \textcolor{green}{tL}$ )
6. ( $\neg \textcolor{blue}{w}$ )

Added clauses:

7. (**tL**  $\vee$  **tR**)
8. (**tL**)

# Street Crosser Agent: Summary

Applying resolution to all possible PERCEPTS and Q (only one) combinations and decisions:

- $\text{PERCEPTS} \equiv (\neg \text{tR}) \wedge (\neg \text{tL}) \rightarrow \text{WALK}$
- $\text{PERCEPTS} \equiv (\text{tR}) \wedge (\neg \text{tL}) \rightarrow \text{DON'T WALK}$
- $\text{PERCEPTS} \equiv (\neg \text{tR}) \wedge (\text{tL}) \rightarrow \text{DON'T WALK}$
- $\text{PERCEPTS} \equiv (\text{tR}) \wedge (\text{tL}) \rightarrow \text{DON'T WALK}$

allowed our agent to:

- reason and make decisions
- learn: percepts  $\rightarrow$  decision is new knowledge!

# Knowledge Base: But wait...

If I keep adding multiple new PERCEPTS to the knowledge base KB, for example:

$$\text{PERCEPTS1} \equiv (\neg \text{tR}) \wedge (\neg \text{tL})$$

$$\text{PERCEPTS2} \equiv (\text{tR}) \wedge (\text{tL})$$

I may end up with a contradiction in my KB, right?



# Knowledge-based Agents

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

KBBEFORE

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*

CURRENTKB

new *percept*

$\text{CURRENTKB} \Leftrightarrow \text{KBBEFORE} \wedge \text{percept}$

# Knowledge-based Agents

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

KBBEFORE

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*

CURRENTKB

"time stamps"

new *percept*

$\text{CURRENTKB} \Leftrightarrow \text{KBBEFORE} \wedge \text{new percept}$

# Automated PL Resolution:Pseudocode

**function** PL-RESOLUTION( $KB, \alpha$ ) **returns** *true* or *false*

**inputs:**  $KB$ , the knowledge base, a sentence in propositional logic  
 $\alpha$ , the query, a sentence in propositional logic

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

$new \leftarrow \{ \}$

**while** *true* **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*

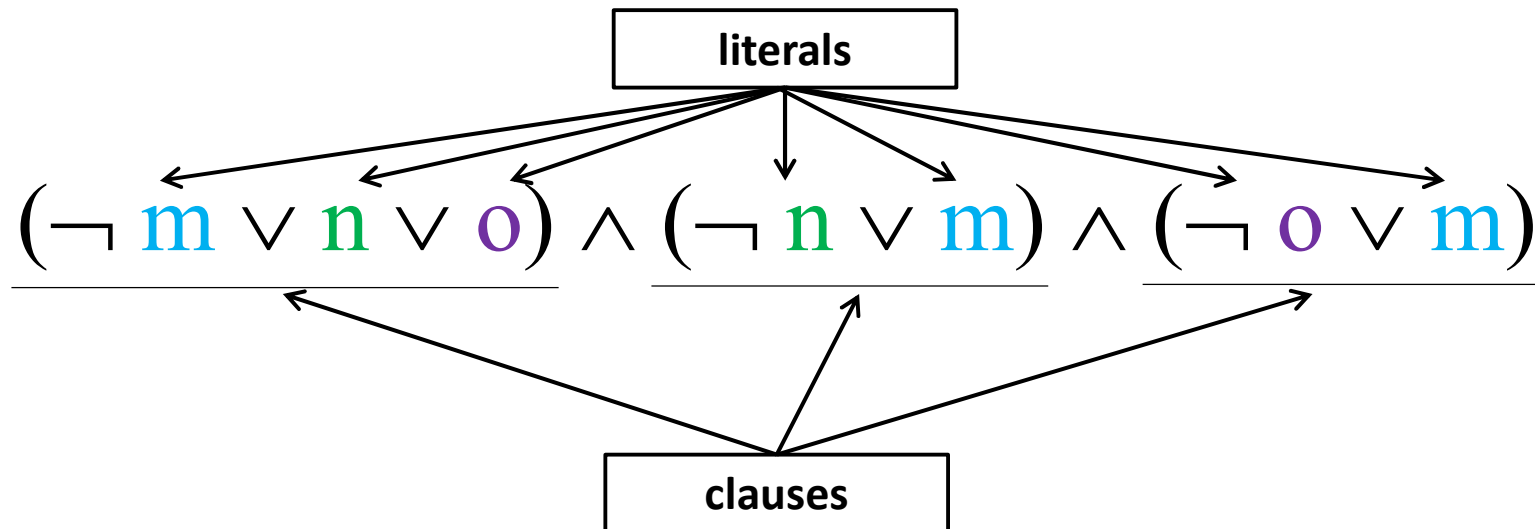
$clauses \leftarrow clauses \cup new$

PL-RESOLVE function will return a set of **ALL possible clauses** obtained by resolving  $C_i$  and  $C_j$ .

# Conjunctive Normal Form (CNF)

Example:

Sentence  $m \Leftrightarrow (n \vee o)$  converted into CNF:



CNF form **enables** (automated) resolution.

# Definite Clauses

A sentence can be called a **definite clause** if and only if it is a **disjunction of literals of which EXACTLY one is positive**. For example:

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

is a definite clause.

This:

$$(x \vee \neg y \vee z)$$

is NOT a definite clause (more than one positive literal)



# Horn Clauses

A sentence can be called a **Horn clause** if and only if it is a **disjunction of literals of which AT MOST one is positive**. For example:

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

is a Horn clause. This:

$$(x \vee \neg y \vee z)$$

is NOT a Horn clause. However, this:

$$(\neg d \vee \neg e \vee \neg f)$$

is a Horn clause (goal clause  $\rightarrow$  no positive literals).

# Definite / Horn Clauses: Why Bother?

Reasons to use definite / Horn clauses:

- resolution of two Horn clauses, yields a Horn clause
- definite clauses can be rewritten as implications:
$$(\neg p \vee \neg q \vee r) \equiv (p \wedge q) \Rightarrow r$$
- inference with Horn clauses can be realized using two human-friendly (-understandable) algorithms: forward- and backward-chaining
- deciding entailment with Horn clauses is  $O(|KB|)$

# Definite / Horn Clauses: Why Bother?

Reasons to use definite / Horn clauses:

- resolution of two Horn clauses, yields a Horn clause
- definite clauses can be rewritten as implications:
$$(\neg p \vee \neg q \vee r) \equiv (p \wedge q) \Rightarrow r$$
- inference with Horn clauses can be realized using two human-friendly (-understandable) algorithms: forward- and backward-chaining
- **deciding entailment with Horn clauses is  $O(|KB|)$**



# Types of Horn Clauses

## Types of Horn clauses (at most one positive literal):

Type of Horn clause	Disjunction form	Implication form	Read in English as
Definite clause	$(\neg p \vee \neg q \vee \dots \vee \neg t \vee \mathbf{u})$	$(p \wedge q \wedge \dots \wedge t) \Rightarrow \mathbf{u}$	assume that, if $p$ and $q$ and ... and $t$ all hold, then also $\mathbf{u}$ holds Rules   If .... then
Fact / Unit Clause	$\mathbf{u}$	$\top \Rightarrow \mathbf{u}$	assume that $\mathbf{u}$ holds
Goal clause	$(\neg p \vee \neg q \vee \dots \vee \neg t)$	$(p \wedge q \wedge \dots \wedge t) \Rightarrow \perp$	show that $p$ and $q$ and ... and $t$ all hold

$$(\neg p \vee \neg q \vee \dots \vee \neg t \vee \mathbf{u}) \equiv \neg(p \wedge \neg q \wedge \dots \wedge \neg t) \vee \mathbf{u}$$

Because (Implication elimination reversed)  $\neg \mathbf{a} \vee \mathbf{b} \equiv \mathbf{a} \Rightarrow \mathbf{b}$ :

$$\neg(p \wedge \neg q \wedge \dots \wedge \neg t) \vee \mathbf{u} \equiv (p \wedge \neg q \wedge \dots \wedge \neg t) \Rightarrow \mathbf{u}$$

**Also:**  $(\neg p \vee \neg q \vee \dots \vee \neg t \vee \mathbf{u}) \equiv (\text{head/consequence} \vee \text{body/premise})$

# Definite Clause and Modus Ponens

## Modus Ponens

$P \Rightarrow Q$

$Q$

$\therefore Q$

## Modus Ponens (textbook)

$(P \Rightarrow Q), (Q)$

$(Q)$

# Definite Clause and Modus Ponens

## Modus Ponens

$$(p \wedge \neg q \wedge \dots \wedge \neg t) \Rightarrow u$$

$u$

$\therefore u$

## Modus Ponens (textbook)

$$((p \wedge \neg q \wedge \dots \wedge \neg t) \Rightarrow u), (u)$$

$(u)$

# Forward Chaining Algorithm

Entailment can be verified with Forward Chaining:

- set up your Knowledge Base KB
- set up your query Q
- start with known facts (say A and B):
  - A and B are automatically considered “inferred”
  - are they a part of some implication  $A \wedge B \Rightarrow X$ ?
  - if yes, X is now considered “inferred”
- Repeat until:
  - Q is “inferred”, or
  - no further inferences can be made

# Forward Chaining: Pseudocode

**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 initially the number of symbols in clause  $c$ 's premise

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

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

**while**  $queue$  is not empty **do**

$p \leftarrow \text{POP}(queue)$

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

**if**  $inferred[p] = \text{false}$  **then**

$inferred[p] \leftarrow \text{true}$

**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  $queue$

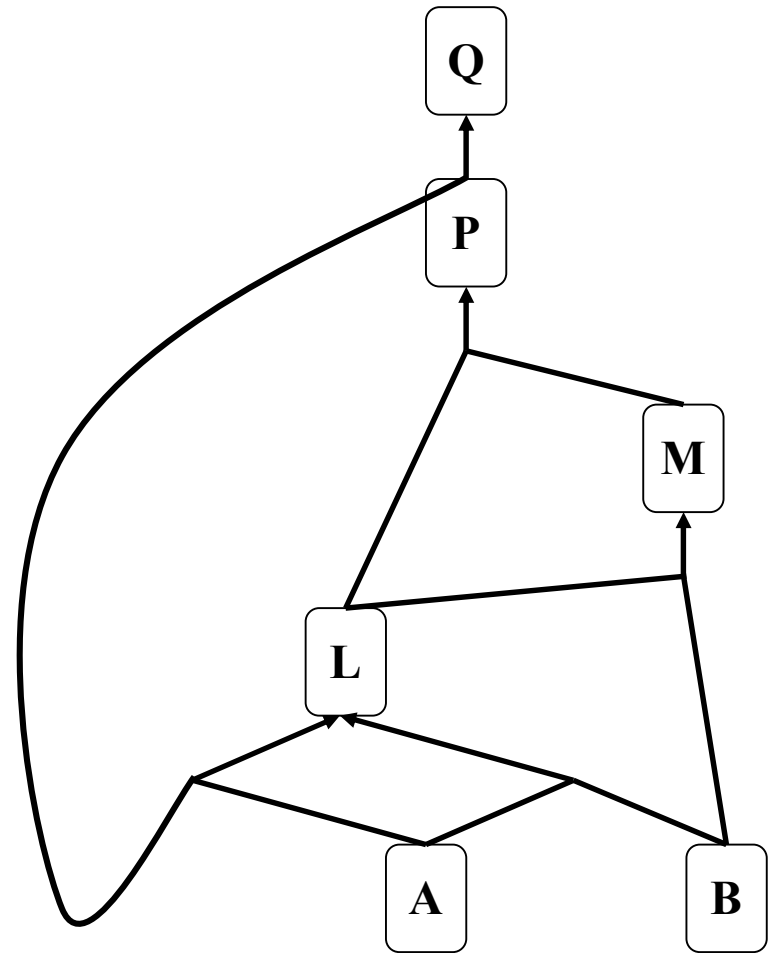
**return** *false*

# Forward Chaining Algorithm

Knowledge Base 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

Inferred



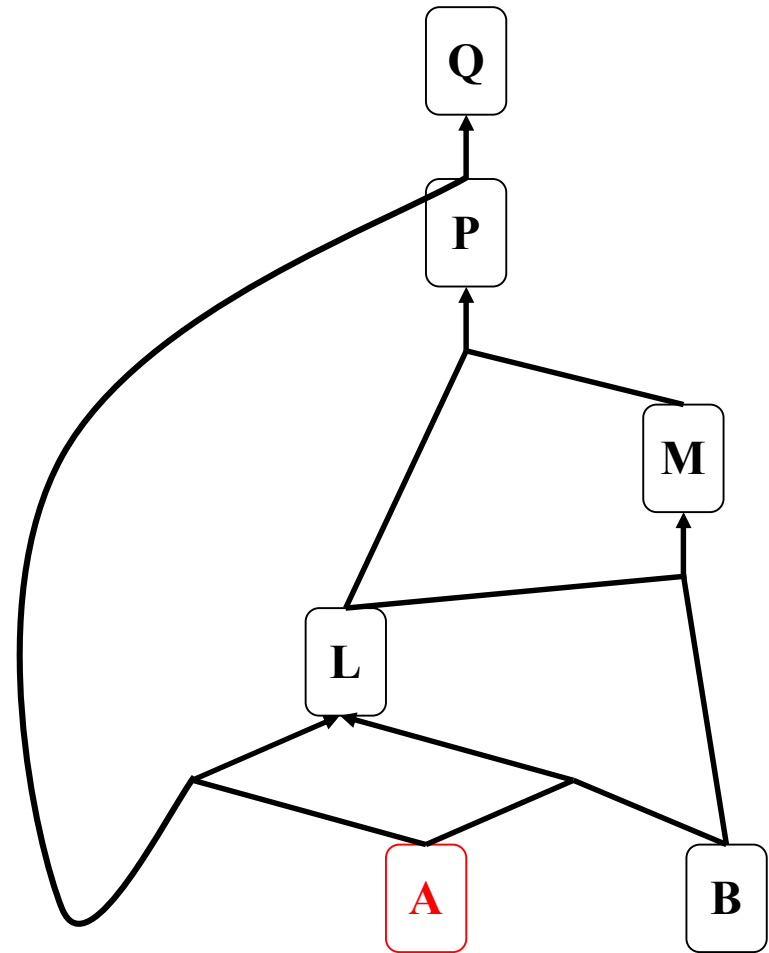
# Forward Chaining Algorithm

Knowledge Base 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

Inferred

**A (because it is a fact)**



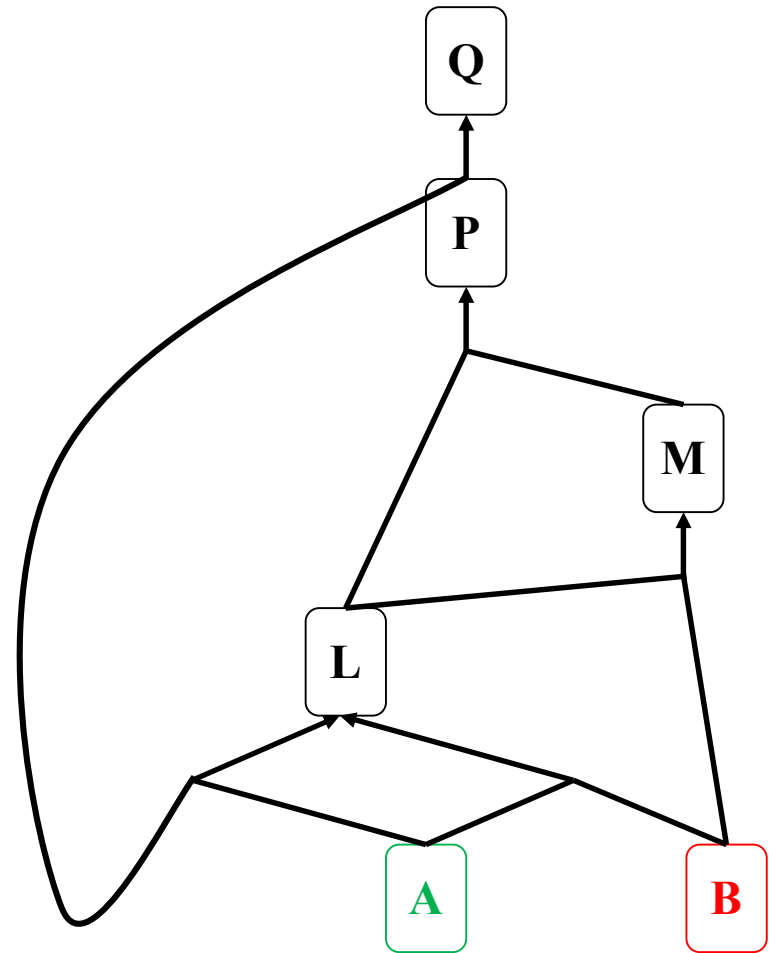
# Forward Chaining Algorithm

Knowledge Base 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**

Inferred:

**A**  
**B (because it is a fact)**





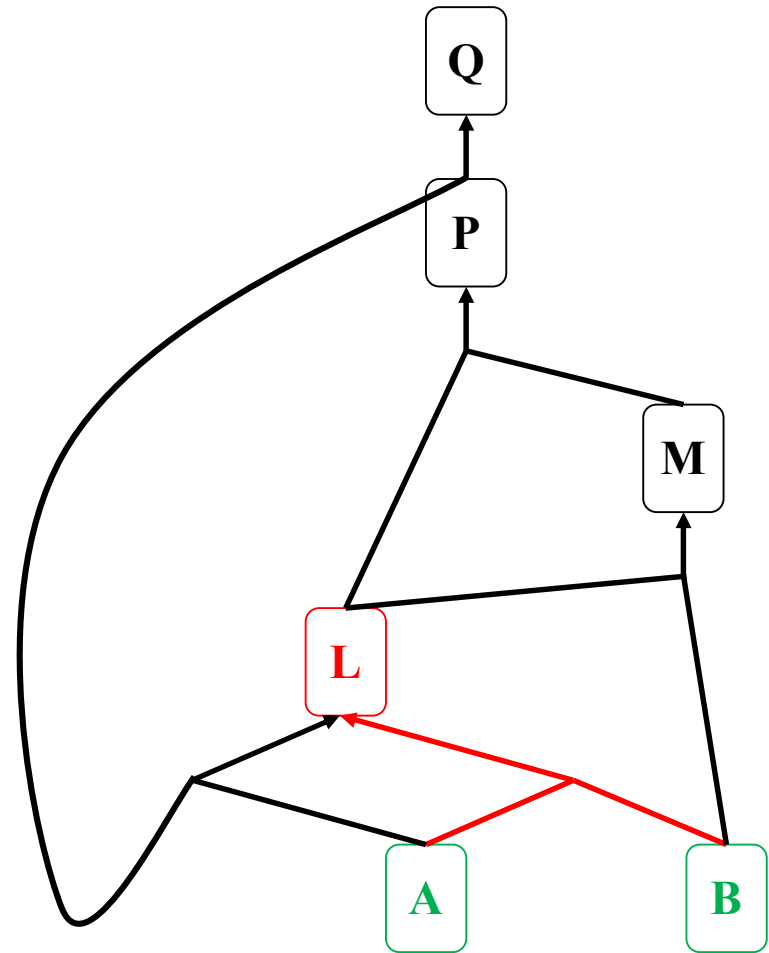
# Forward Chaining Algorithm

Knowledge Base KB:

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

Inferred:

$A$   
 $B$   
 $L$  (because  $A \wedge B \Rightarrow L$ )



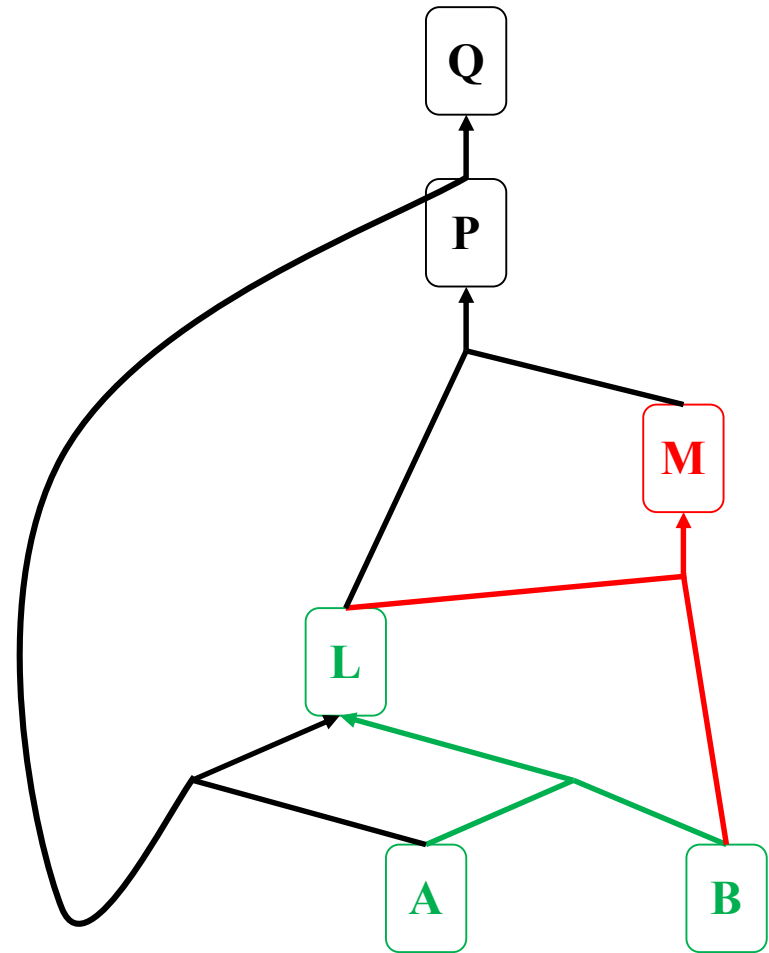
# Forward Chaining Algorithm

Knowledge Base KB:

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

Inferred:

$A$   
 $B$   
 $L$  (because  $A \wedge B \Rightarrow L$ )  
 $M$  (because  $B \wedge L \Rightarrow M$ )



# Forward Chaining Algorithm

Knowledge Base 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

Inferred:

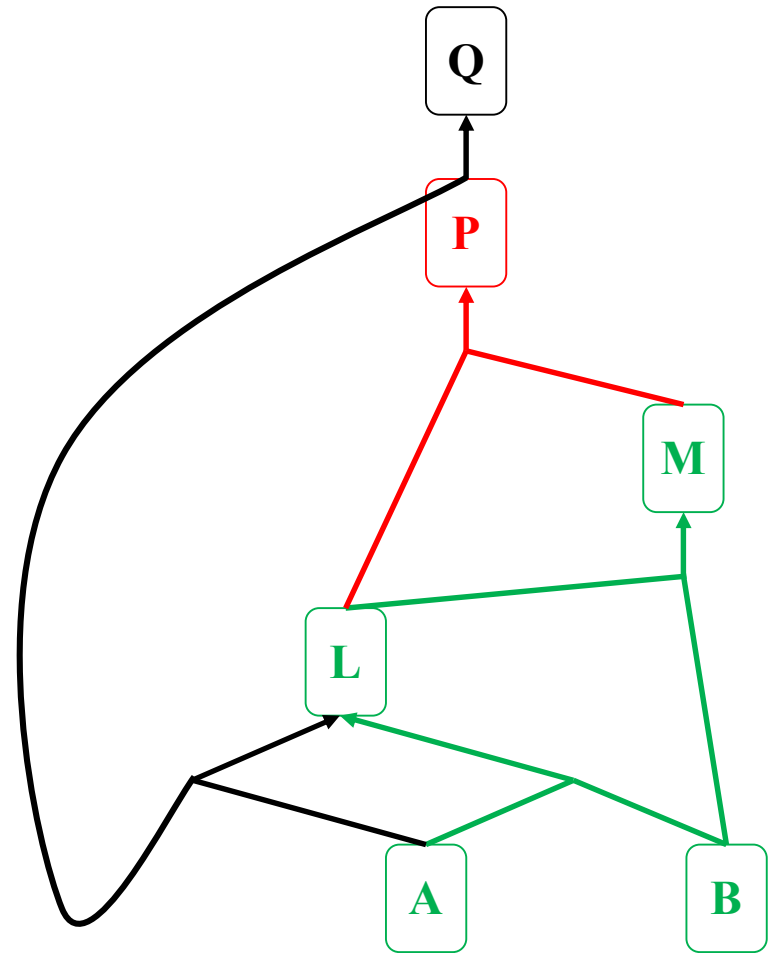
A

B

L (because  $A \wedge B \Rightarrow L$ )

M (because  $B \wedge L \Rightarrow M$ )

P (because  $L \wedge M \Rightarrow P$ )



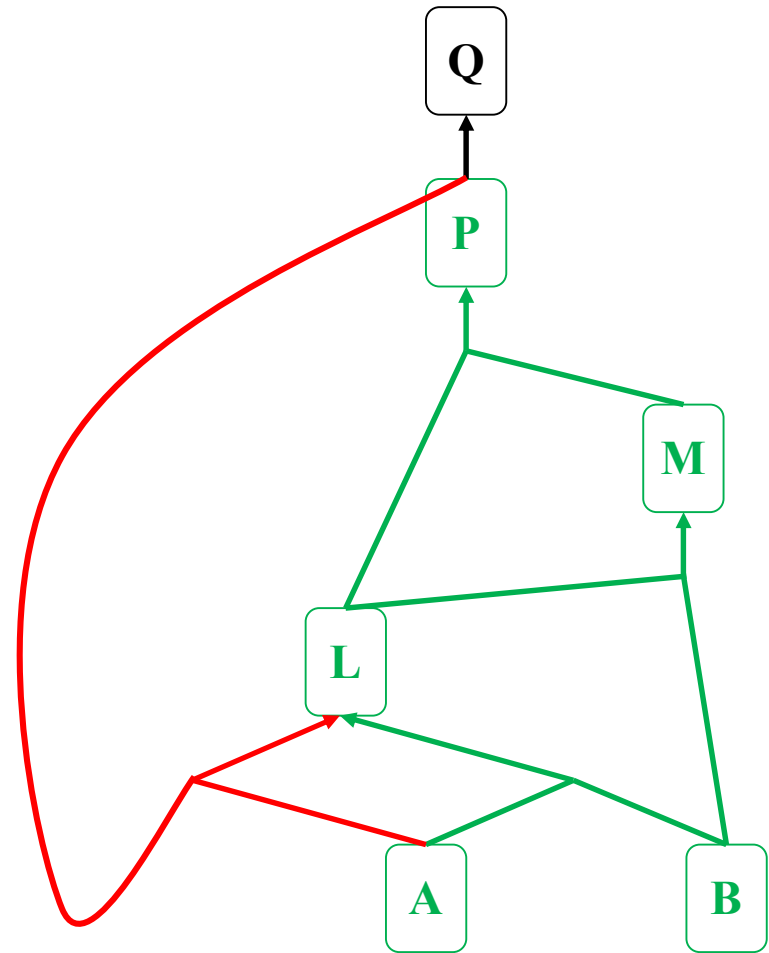
# Forward Chaining Algorithm

Knowledge Base KB:

$P \Rightarrow Q$   
 $L \wedge M \Rightarrow P$   
 $B \wedge L \Rightarrow M$   
 $A \wedge P \Rightarrow L$  (note: L is already inferred)  
 $A \wedge B \Rightarrow L$   
A  
B

Inferred:

A  
B  
L (because  $A \wedge B \Rightarrow L$ )  
M (because  $B \wedge L \Rightarrow M$ )  
P (because  $L \wedge M \Rightarrow P$ )



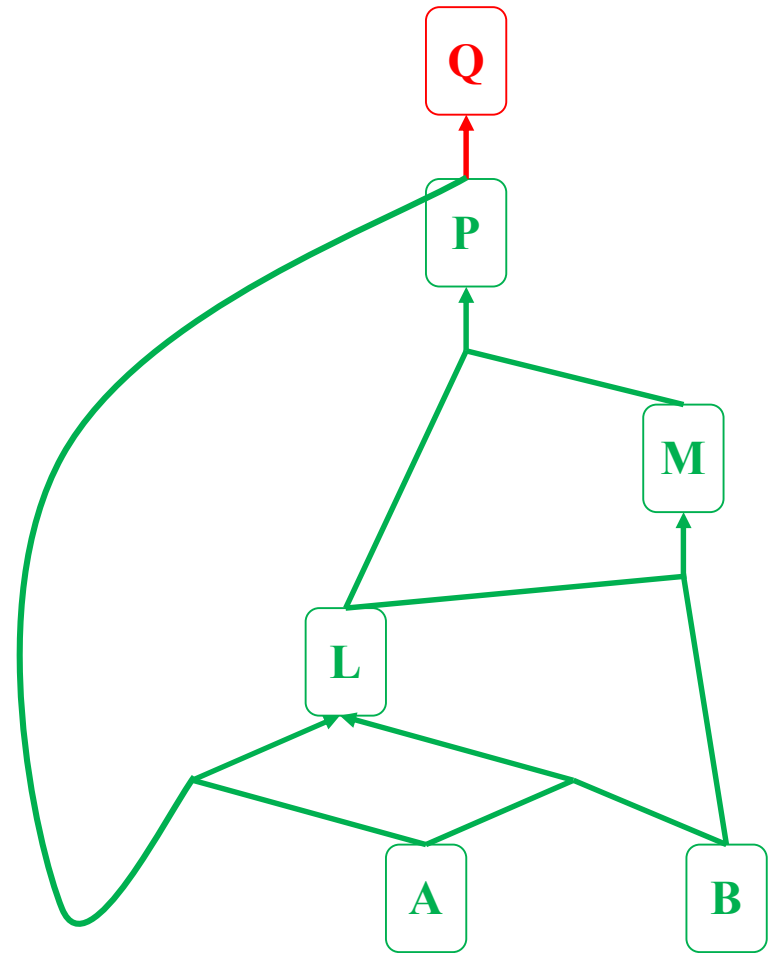
# Forward Chaining Algorithm

Knowledge Base 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

Inferred:

A  
B  
L (because  $A \wedge B \Rightarrow L$ )  
M (because  $B \wedge L \Rightarrow M$ )  
P (because  $L \wedge M \Rightarrow P$ )  
 $Q$  (because  $P \Rightarrow Q$ )



# Forward Chaining Algorithm

Knowledge Base KB:

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

Inferred:

$A$   
 $B$   
 $L$  (because  $A \wedge B \Rightarrow L$ )  
 $M$  (because  $B \wedge L \Rightarrow M$ )  
 $P$  (because  $L \wedge M \Rightarrow P$ )  
 $Q$  (because  $P \Rightarrow Q$ )  
 $Q$  is inferred, therefore KB entails  $Q$

