CS 480

Introduction to Artificial Intelligence

October 5th, 2021

Announcements / Reminders

- **Written Assignment #02:**
  - **due: TONIGHT, 11:00 PM CST**
- **Programming Assignment #01:**
  - **due: ~~March 6th~~ March 13th, 11:00 PM CST**
- **Grading TA assignment:**

  https://docs.google.com/spreadsheets/d/1avK4P4MDjKZQceG82mSZd0wkYEDH07_DpQqYJHDQctw/edit?usp=sharing

Plan for Today

- **Propositional logic and inference**

Logical Entailment with KB Agents

But we could ask the following question:

"Which ACTIONs follow from CURRENTKB?"

$(\text{KBBEFORE} \wedge \text{percept}) \models$

- ACTION_1 follows from
- ACTION_2 does NOT follow
- ...
- ACTION_{m-1} follows from
- ACTION_m follows from

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:

$$(\text{KBBEFORE} \wedge \text{percept}) \models \text{ACTION}$$ 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 an right. KB agent cannot cross if there is ANY traffic (ignore the STOP sign).

KB Agent: Should I Stay or Should I Go

Sensor

PERCEPT:

Knowledge Base and Inference System

Actuator

ACTION:

KB Agent: Street Crosser

Environment: Street

KB Agent: Should I Stay or Should I Go

Sensor

TRAFFIC LEFT / RIGHT

Knowledge Base and Inference System

Actuator

WALK OR STAY

KB Agent: Street Crosser

Environment: Street

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: Knowledge Base KB

English:

A: “Walk if and only if there is NO traffic coming from the left AND NO traffic coming from the right.”

or

B: “DON’T walk if and only if there is traffic coming from the left OR traffic coming from the right.”

Street Crosser: Knowledge Base KB

English and Propositional Logic:

A: "Walk if and only if there is NO traffic coming from the left AND NO traffic coming from the right."

$$\mathrm{walk} \Leftrightarrow (\neg \mathrm{trafficLeft} \wedge \neg \mathrm{trafficRight})$$

or

B: "DON'T walk if and only if there is traffic coming from the left OR traffic coming from the right."

$$\neg \mathrm{walk} \Leftrightarrow (\mathrm{trafficLeft} \vee \mathrm{trafficRight})$$

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

Street Crosser: Convert KB to CNF

Variant B: $KB \equiv \neg walk \Leftrightarrow (trafficLeft \vee trafficRight)$

$$(\neg walk \Rightarrow (trafficLeft \vee trafficRight)) \wedge ((trafficLeft \vee trafficRight) \Rightarrow \neg walk)$$

by Biconditional Elimination

$$(\neg(\neg walk) \vee (trafficLeft \vee trafficRight)) \wedge (\neg(trafficLeft \vee trafficRight) \vee \neg walk)$$

by Implication Elimination

$$(walk \vee (trafficLeft \vee trafficRight)) \wedge (\neg(trafficLeft \vee trafficRight) \vee \neg walk)$$

by Double Negation Elimination

$$(walk \vee (trafficLeft \vee trafficRight)) \wedge ((\neg trafficLeft \wedge \neg trafficRight) \vee \neg walk)$$

by De Morgan's Rule

$$(walk \vee (trafficLeft \vee trafficRight)) \wedge ((\neg trafficLeft \vee \neg walk) \wedge (\neg trafficRight \vee \neg walk))$$

by Distributivity Rule

$$(walk \vee trafficLeft \vee trafficRight) \wedge (\neg trafficLeft \vee \neg walk) \wedge (\neg trafficRight \vee \neg walk)$$

remove extraneous parentheses

Should I Go: Proof by Resolution

We have our knowledge base KB in CNF ready:

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

Let's rename propositional variables to simplify:

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

Assume that <u>traffic is coming from both left and right</u> (percepts):

$$\mathrm{PERCEPTS} \equiv (\mathrm{tR}) \wedge (\mathrm{tL})$$

Let's add (TELL) $\mathrm{PERCEPTS}$ to the Knowledge Base KB:

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

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

Our query Q is "Should I (choose action) walk?":

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

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

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

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

Should I Go: Proof by Resolution

Prove:

$$KB_N \land \neg Q \equiv$$

$$(w \lor tL \lor tR)_1 \land (\neg tL \lor \neg w)_2 \land (\neg tR \lor \neg w)_3 \land (tR)_4 \land (tL)_5 \land (\neg w)_6$$

Should I Go: Proof by Resolution

Prove:

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

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

Known clauses:

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

Added clauses:

Proof by Resolution: Example

Prove:

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

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

Resolution applied to clauses 1 and 6

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

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

Known clauses:

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

Added clauses:

7. $(tL \vee tR)$

Proof by Resolution: Example

Prove:

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

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

Resolution applied to clauses 2 and 5

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

Produces a clause ($\neg w$), but we already have it (6). Don't add it to the list.

Known clauses:

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

Added clauses:

7. $(tL \vee tR)$

Proof by Resolution: Example

Prove:

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

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

Resolution applied to clauses 3 and 4

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

Produces a clause ($\neg w$), but we already have it (6). Don't add it to the list.

Known clauses:

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

Added clauses:

7. $(tL \vee tR)$

Proof by Resolution: Example

Prove:

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

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

Resolution applied to clauses 2 and 7

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

Produces a new clause ($\neg w \vee tR$). We can add it to the list as clause (8).

Known clauses:

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

Added clauses:

7. $(tL \vee tR)$
8. $(\neg w \vee tR)$

Proof by Resolution: Example

Prove:

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

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

Resolution applied to clauses 1 and 8

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

Produces a clause ($tL \vee tR$), but we already have it (7). Don't add it to the list.

Known clauses:

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

Added clauses:

7. $(tL \vee tR)$
8. $(\neg w \vee tR)$

Proof by Resolution: Example

Prove:

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

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

Resolution applied to clauses 3 and 8

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

Produces a clause ($\neg w$), but we already have it (6). Don't add it to the list.

Known clauses:

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

Added clauses:

7. $(tL \vee tR)$
8. $(\neg w \vee tR)$

Proof by Resolution: Example

Prove:

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

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

Resolution applied to clauses 3 and 7

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

Produces a new clause ($\neg w \vee tL$). We can add it to the list as clause (9).

Known clauses:

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

Added clauses:

7. $(tL \vee tR)$
8. $(\neg w \vee tR)$
9. $(\neg w \vee tL)$

Proof by Resolution: Example

Prove:

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

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

Resolution applied to clauses 2 and 9

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

Produces a clause ($\neg w$), but we already have it (6). Don't add it to the list.

Known clauses:

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

Added clauses:

7. $(tL \vee tR)$
8. $(\neg w \vee tR)$
9. $(\neg w \vee tL)$

Proof by Resolution: Example

Prove:

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

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

At this point, we tried to resolve all promising clause pairs, but we have not reached an empty clause $\rightarrow$ KB **does NOT entail** Q.

Given PERCEPTS: $(tR) \wedge (tL)$

we should NOT apply action walk (w) **and stay.**

Known clauses:

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

Added clauses:

7. $(tL \vee tR)$
8. $(\neg w \vee tR)$
9. $(\neg w \vee tL)$

Should I Go: Proof by Resolution

We have our knowledge base KB in CNF ready:

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

Let's rename propositional variables to simplify:

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

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

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

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

$$\mathrm{KB}_N \equiv \mathrm{KB} \wedge \mathrm{PERCEPTS}$$

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

Our query Q is "Should I (choose action) walk?":

$Q \equiv w$ **(and in negated form:** $\neg Q \equiv \neg w$**)**

To test / prove entailment I want to prove that $\mathrm{KB}_N \wedge \neg Q$ **is true:**

$$\mathrm{KB}_N \wedge \neg Q \equiv (w \vee tL \vee tR) \wedge (\neg tL \vee \neg w) \wedge (\neg tR \vee \neg w) \wedge (\neg tR) \wedge (\neg tL) \wedge (\neg w)$$

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

Should I Go: Proof by Resolution

Prove:

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

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

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:

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

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

Resolution applied to clauses 1 and 6

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

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

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:

7. $(tL \vee tR)$

Proof by Resolution: Example

Prove:

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

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

Resolution applied to clauses 4 and 7

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

Produces a new clause (tL). We can add it to the list as clause (8).

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:

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

Proof by Resolution: Example

Prove:

$$KB_N \land \neg Q \equiv$$

$$(w \lor tL \lor tR)_1 \land (\neg tL \lor \neg w)_2 \land (\neg tR \lor \neg w)_3 \land (\neg tR)_4 \land (\neg tL)_5 \land (\neg w)_6$$

Resolution applied to clauses 5 and 8

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

Produces an empty clause / contradiction. Stop.

Known clauses:

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

Added clauses:

7. $(tL \lor tR)$
8. (tL)

Proof by Resolution: Example

Prove:

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

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

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

Given PERCEPTS: $(\neg tR) \wedge (\neg tL)$

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

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:

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

Street Crosser Agent: Summary

Applying resolution to all possible $\mathrm{PERCEPTS}$ **and** Q **(only one) combinations and decisions:**

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

allowed our agent to:

- **reason and make decisions**
- **learn: percepts → decision is new knowledge!**

Knowledge Base: But wait...

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

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

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

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

CURRENTKB

new percept

CURRENTKB ⇔ KBBEFORE ∧ percept

Knowledge-based Agents

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

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

KBBEFORE

"time stamps"

CURRENTKB

new percept

CURRENTKB ⇔ KBBEFORE ∧ percept

Automated PL Resolution:Pseudocode

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 $KB \wedge \neg\alpha$
  $new \leftarrow \{\}$
  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:**

literals

$$(\neg m \vee n \vee o) \wedge (\neg n \vee m) \wedge (\neg o \vee m)$$

clauses

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:

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

is a Horn clause. This:

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

is NOT a Horn clause. However, this:

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

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	$(\neg p \vee \neg q \vee \ldots \vee \neg t \vee u)$	$(p \wedge q \wedge \ldots \wedge t) \Rightarrow u$	assume that, if p and q and ... and t all hold, then also u holds Rules | If .... then
Fact / Unit Clause	u	$T \Rightarrow u$	assume that u holds
Goal clause	$(\neg p \vee \neg q \vee \ldots \vee \neg t)$	$(p \wedge q \wedge \ldots \wedge t) \Rightarrow \perp$	show that p and q and ... and t all hold

$$(\neg p \vee \neg q \vee \ldots \vee \neg t \vee u) \equiv \neg(p \wedge \neg q \wedge \ldots \wedge \neg t) \vee u$$

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

$$\neg(p \wedge \neg q \wedge \ldots \wedge \neg t) \vee u \equiv (p \wedge \neg q \wedge \ldots \wedge \neg t) \Rightarrow u$$

Also: $(\neg p \vee \neg q \vee \ldots \vee \neg t \vee u) \equiv$ (head/consequence $\vee$ body/premise)

Definite Clause and Modus Ponens

Modus Ponens

$P \Rightarrow Q$

Q

$\therefore\ Q$

Modus Ponens (textbook)

$$\frac{(P \Rightarrow Q),\ (Q)}{(Q)}$$

Definite Clause and Modus Ponens

Modus Ponens

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

u

$\therefore\ u$

Modus Ponense (textbook)

$$\frac{((p \wedge \neg q \wedge \ldots \wedge \neg t) \Rightarrow u),\ (u)}{(u)}$$

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** $\mathrm{A} \wedge \mathrm{B} \Rightarrow \mathrm{X}$**?**
  - **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

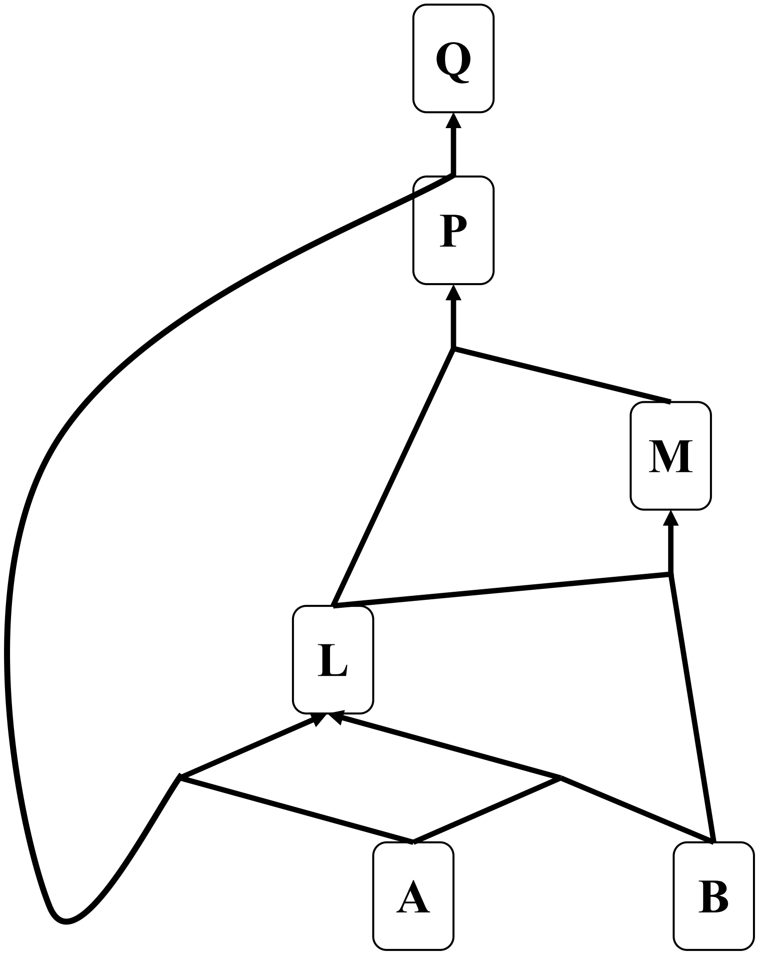
  while $queue$ is not empty **do**
    $p \leftarrow$ POP($queue$)
    if $p = q$ **then return** *true*
    if $inferred[p] = false$ **then**
      $inferred[p] \leftarrow true$
      for each clause c in KB where p is in c.PREMISE **do**
        decrement $count[c]$
        if $count[c] = 0$ **then** add c.CONCLUSION to $queue$
  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



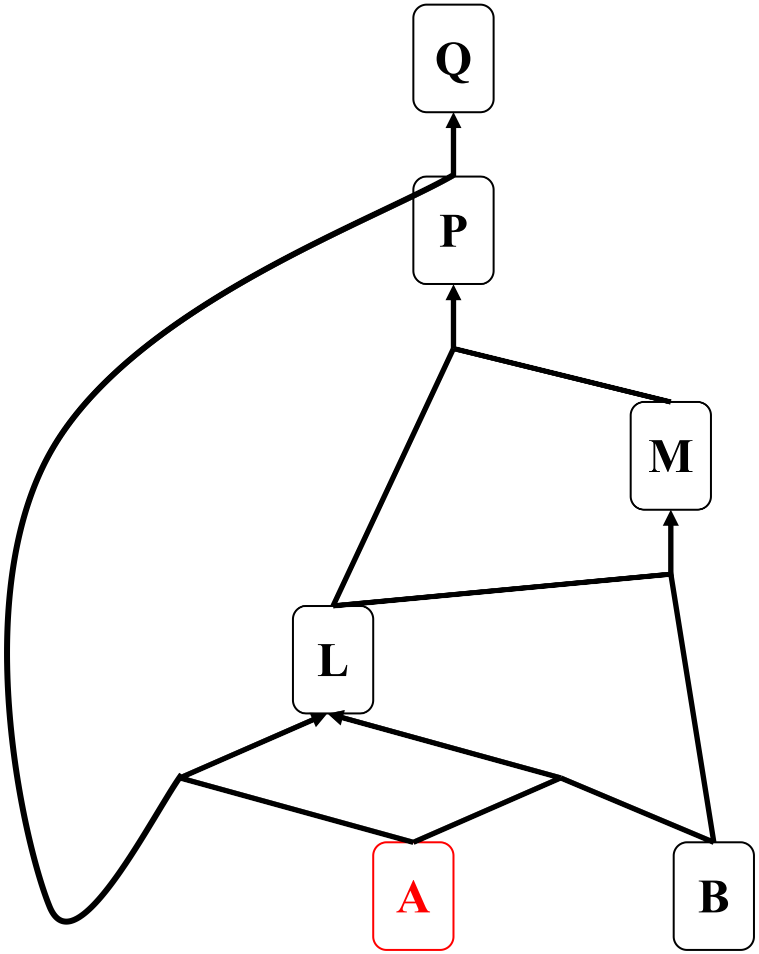
Forward Chaining Algorithm

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:

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

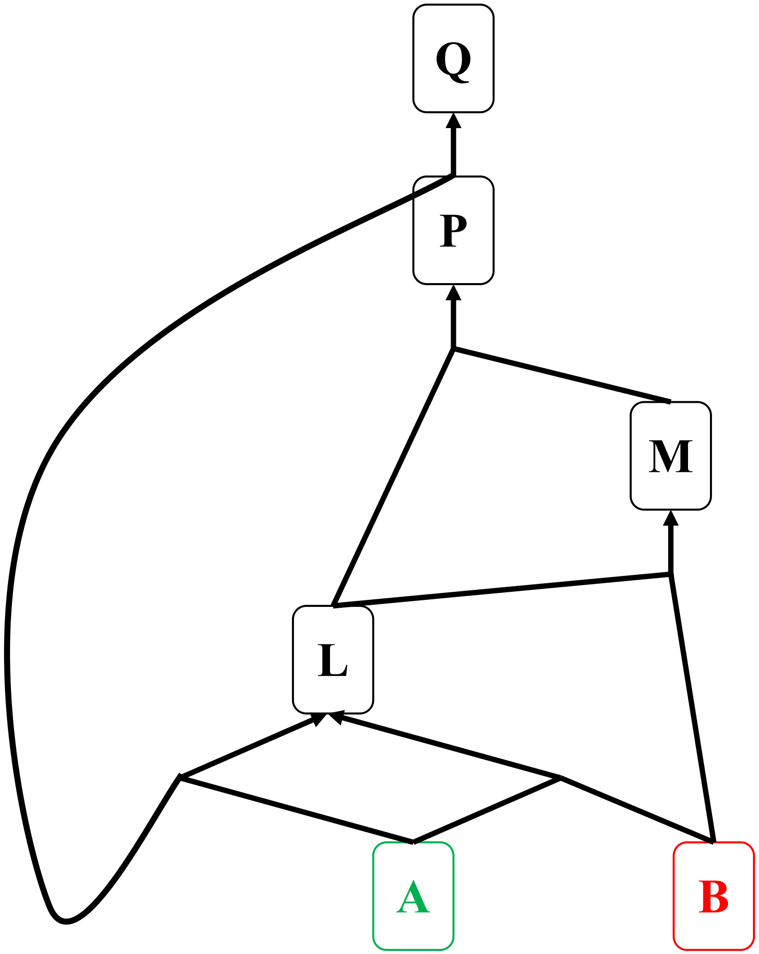
A

B

Inferred:

A

B (because it is a fact)



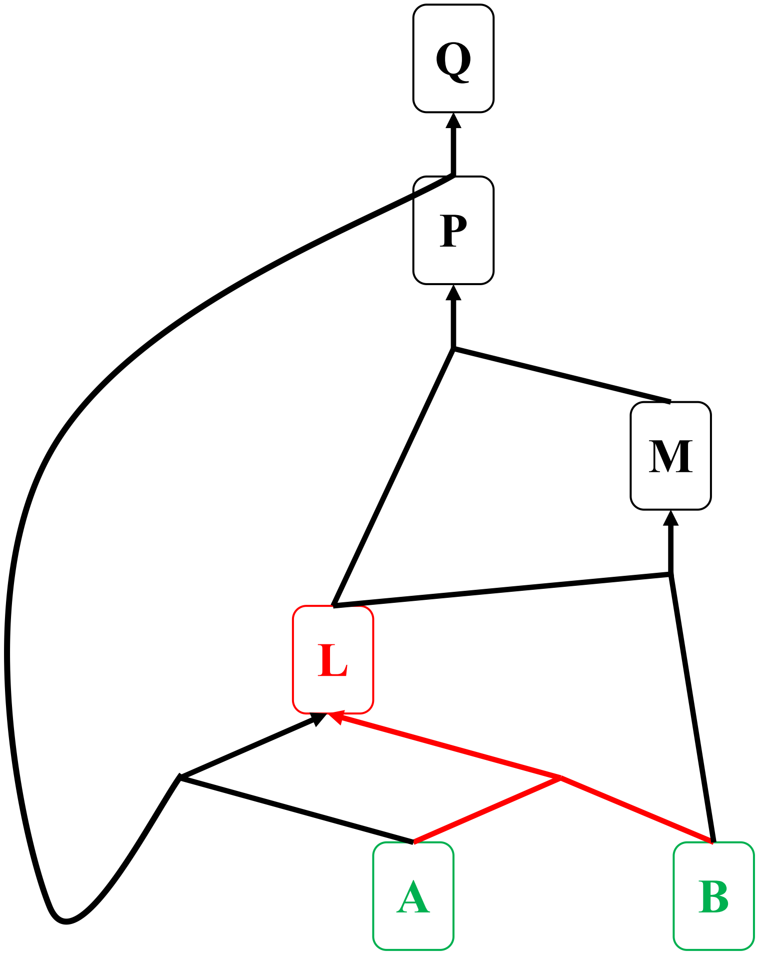
Forward Chaining Algorithm

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 $A \wedge B \Rightarrow L$)



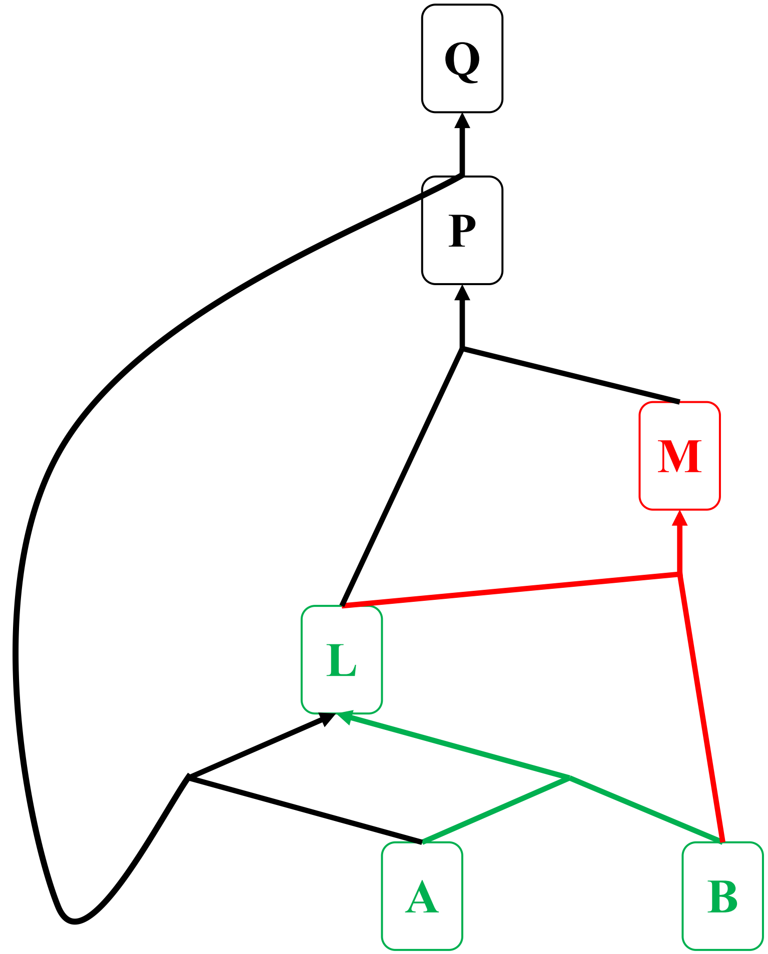
Forward Chaining Algorithm

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 $B \wedge L \Rightarrow M$)



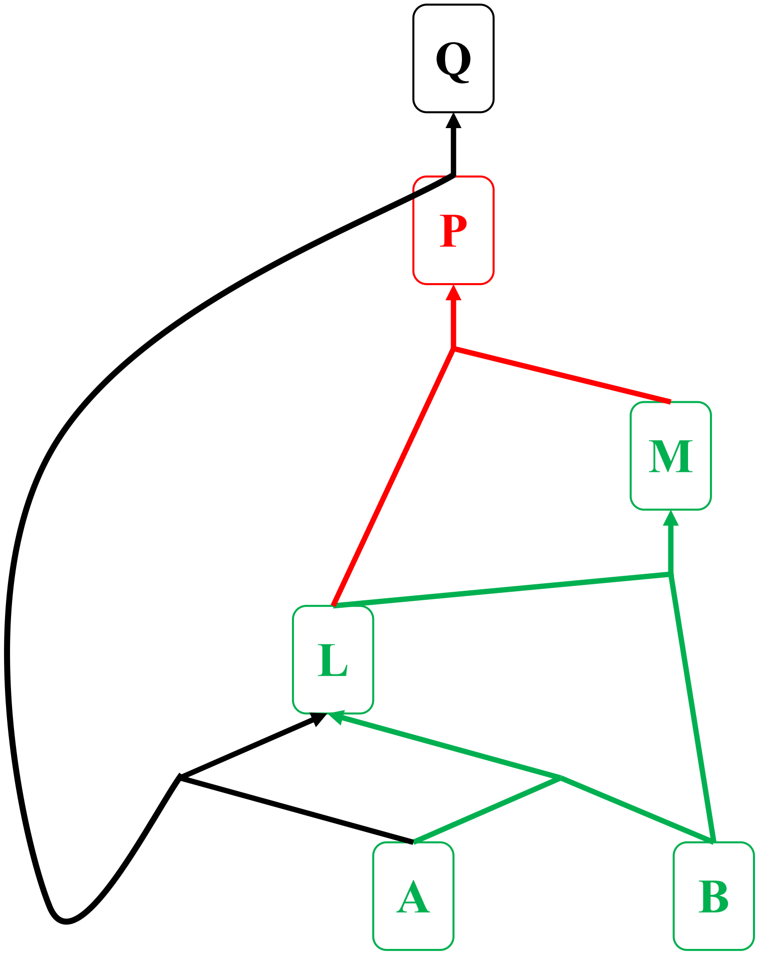
Forward Chaining Algorithm

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 $L \wedge M \Rightarrow P$)



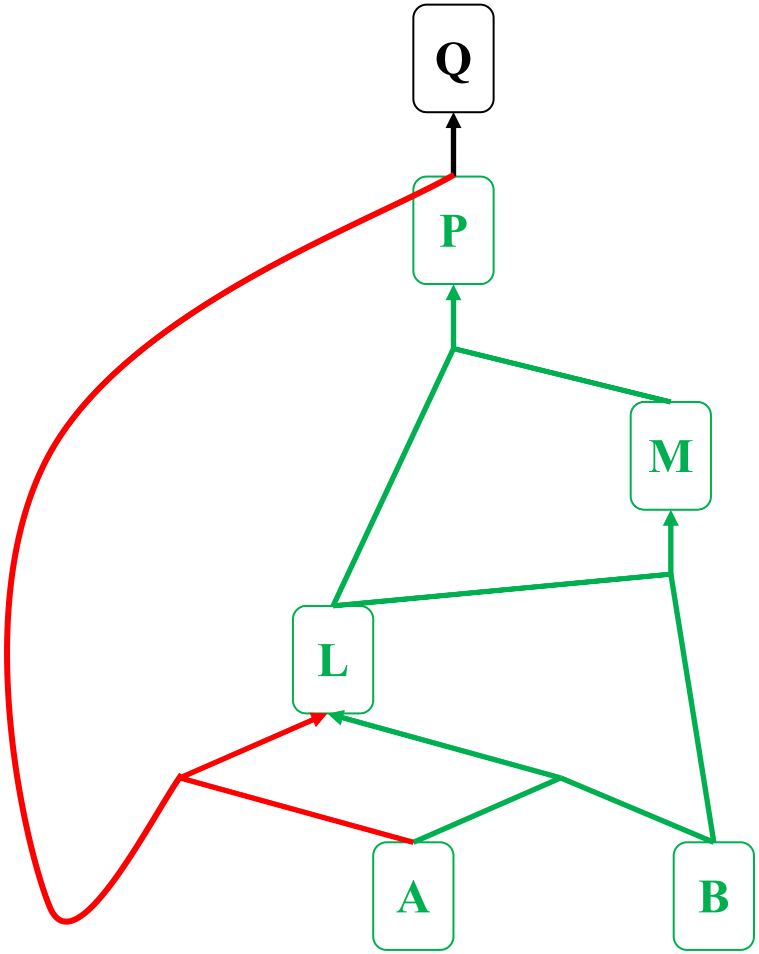
Forward Chaining Algorithm

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 $L \wedge M \Rightarrow P$)



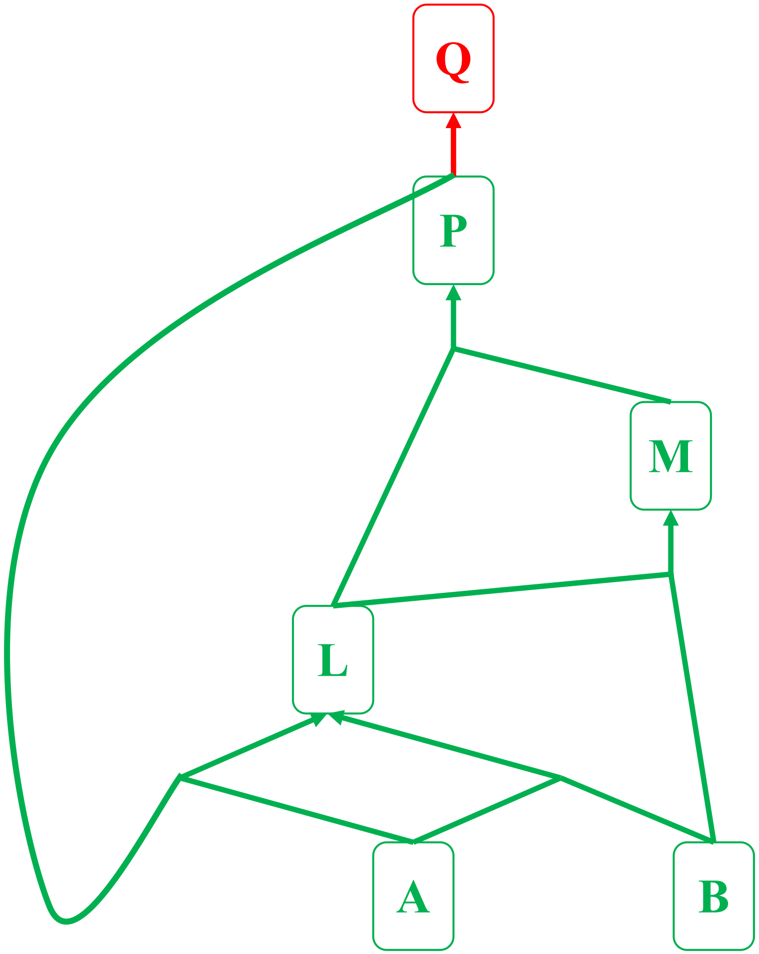
Forward Chaining Algorithm

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 <u>KB entails Q</u>