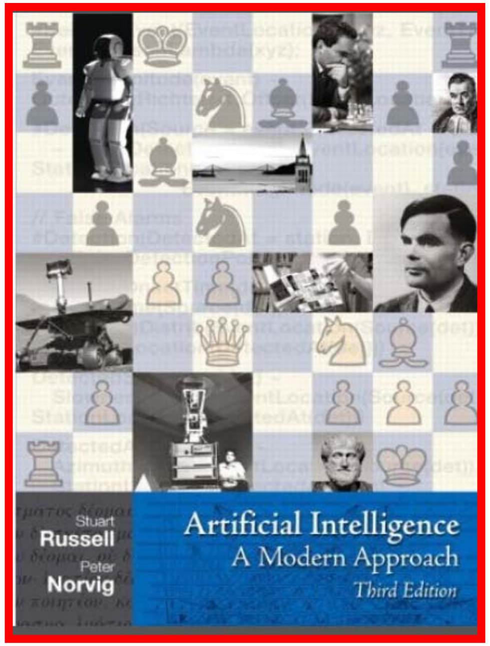


(Chapter-7)

Logical Agents---Resolution

Yanmei Zheng

From KB. derive α

from [KB,- -a), derive False

[KB, a) is unsatisfiable

using resolution inference rule

Resolution{7.5.2}

- is the following inference rule sound?

$$\frac{A \vee \neg B,\; B \vee C}{A \vee C}$$

- Resolution rule

Complementary Literals [7.5.2)

- A literal is a either an atomic sentence or the negated atomic sentence, e.g.:

  P, ¬P

- Two literals are complementary if one is the negation of the other, e.g.:

  P and ¬P

CNF (7.5.2)

- Clause

  disjunction of literals

  E.g., $A \vee \neg B$
- Conjunctive Normal Form

  conjunction of disjunctions of literals

  E.g.,$(A \vee \neg B) \wedge (B \vee C)$:

Basic intuition, resolve B, $\neg B$ to $A \vee C$

CNF:why?how?

Converting wffs to Conjunctions of Clauses

- Resolution is a powerful tool for algorithmic inference, but we can only apply it to conjunctions of clauses (conjunctive normal form, CNF).

- So is there a way to convert any wff into such a conjunction of clauses?

- Fortunately, there is such a way, allowing us to apply resolution to any wff.

Conversion to CNF

Conjunctive Normal Form (CNF)

conjunction of disjunctions of literals: clauses

E.g., $(A \vee \neg B) \wedge (B \vee \neg C \vee \neg D)$

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

1. Eliminate $\Leftrightarrow$, replacing $\alpha \Leftrightarrow \beta$ with $(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)$.

   $(B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})$
2. Eliminate $\Rightarrow$, replacing $\alpha \Rightarrow \beta$ with $\neg\alpha \vee \beta$.

   $(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge (\neg(P_{1,2} \vee P_{2,1}) \vee B_{1,1})$
3. Move $\neg$ inwards using de Morgan's rules and double-negation:

   $(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge ((\neg P_{1,2} \wedge \neg P_{2,1}) \vee B_{1,1})$
4. Apply distributivity law ($\wedge$ over $\vee$) and flatten:

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

Converting wffs to Conjunctions of Clauses

- **Example:** $\neg(P \rightarrow Q) \vee (R \rightarrow P)$.
- **Step 1:** Eliminate implication operators:
- $\neg(\neg P \vee Q) \vee (\neg R \vee P)$
- **Step 2:** Reduce the scopes of $\neg$ operators by using DeMorgan's laws and eliminating double $\neg$ operators:
- $(P \wedge \neg Q) \vee (\neg R \vee P)$
- **Step 3:** Convert to CNF by using the associative and distributive laws:
- $(P \vee \neg R \vee P) \wedge (\neg Q \vee \neg R \vee P)$, and then
- $(P \vee \neg R) \wedge (\neg Q \vee \neg R \vee P)$

Resolution Refutations

- Resolution is a sound rule of inference, but it is **not complete**.
- For example, $P \wedge R \not\models P \vee R$, but the resolution rule does not allow us to infer $P \vee R$ from {P}, {R}.
- However, we can use resolution to show that the **negation** of $P \vee R$ is **inconsistent** with {P}, {R} and thereby showing that $P \wedge R \models P \vee R$.
- The negation of $P \vee R$ is $\neg P \wedge \neg R$.
- Conjunctively combining all clauses results in $\neg P \wedge \neg R \wedge P \wedge R$.
- This resolves to the empty set, so by contradiction we have indirectly shown that $P \wedge R \not\models P \vee R$.

More Explanation of Resolution Refutations

- **You have a set of hypotheses $h_1, h_2, \ldots, h_n$, and a conclusion c.**

- **Your argument is that whenever all of the $h_1, h_2, \ldots, h_n$ are true, then c is true as well.**

- **In other words, whenever all of the $h_1, h_2, \ldots, h_n$ are true, then $\neg c$ is false.**

- **If and only if the argument is valid, then the conjunction $h_1 \wedge h_2 \wedge \ldots \wedge h_n \wedge \neg c$ is false, because either (at least) one of the $h_1, h_2, \ldots, h_n$ is false, or if they are all true, then $\neg$c is false.**

- **Therefore, if this conjunction resolves to false, we have shown that the argument is valid.**

Resolution

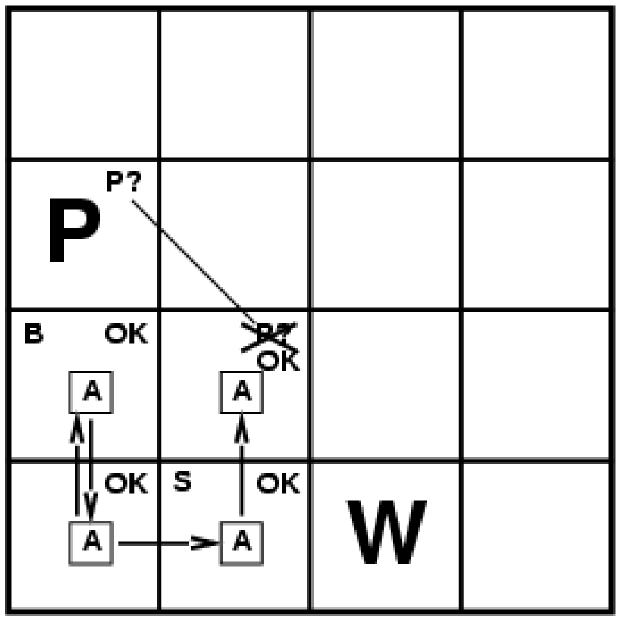
- **Resolution** inference rule (for CNF):

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

where l_i and m_j are complementary literals.

E.g.,
$$\frac{P_{1,3} \vee P_{2,2}, \quad \neg P_{2,2}}{P_{1,3}}$$

- Resolution is sound and complete for propositional logic



Resolution

$$\frac{P, \neg P \vee Q}{Q}$$

$$\frac{P \vee Q, \neg P \vee Q}{Q}$$

$$\frac{\neg P \vee Q, \neg Q \vee R}{\neg P \vee R}$$

$$\frac{P, \neg P}{NIL}$$

Resolution{7.5.2}

- Sound ?

$$\frac{P \vee Q \quad \neg P \vee \neg Q}{NIL}$$

$$\frac{P \vee Q \quad \neg P \vee \neg Q}{Q \vee \neg Q}$$

$$\frac{P \vee Q \quad \neg P \vee \neg Q}{P \vee \neg P}$$

Resolution{7.5.2}

- Sound？

$$\frac{P \vee Q \quad \neg P \vee \neg Q}{NIL}$$ **NO**

$$\frac{P \vee Q \quad \neg P \vee \neg Q}{Q \vee \neg Q}$$ **YES**

$$\frac{P \vee Q \quad \neg P \vee \neg Q}{P \vee \neg P}$$ **YES**

Resolution{7.5.2}

- How to prove KB ⊨ α using resolution?

- consider

  KB ⊨ α iff (KB ∧ ¬α) is unsatisfiable and

$$\frac{P \quad \neg \quad P}{NIL}$$

Resolution Refutations

- **Example:**

- **"Gary is intelligent or a good actor.**
- **If Gary is intelligent, then he can count from 1 to 10.**
- **Gary can only count from 1 to 2.**
- **Therefore, Gary is a good actor."**

- **Propositions:**

- **I: "Gary is intelligent."**
- **A: "Gary is a good actor."**
- **C: "Gary can count from 1 to 10."**

Resolution Refutations

- **Hypotheses:**
  $I \vee A, I \rightarrow C, \neg C$
  In CNF: $(I \vee A) \wedge (\neg I \vee C) \wedge \neg C$
- **Conclusion:** A
- **Conjunction of Clauses for Resolution Refutation:**
- $(I \vee A) \wedge (\neg I \vee C) \wedge \neg C \wedge \neg A$

- **Resolution on A:** $I \wedge (\neg I \vee C) \wedge \neg C$

- **Resolution on I:** $C \wedge \neg C$

- **Resolution on C:** False

- Therefore, the initial set of clauses is inconsistent, and the conclusion is correct: Gary is a good actor.

Resolution Refutations

- **Another Example:**
- **"If Jim visits a pub on Thursday, he is late for work on Friday.**
- **If Jim is late for work on Friday, he has to work during the weekend.**
- **Jim had to work during the weekend.**
- **Therefore, Jim visited a pub on Thursday."**

- **Propositions:**

- **T: "Jim visits a pub on Thursday."**
- **F: "Jim is late for work on Friday."**
- **W: "Jim has to work during the weekend."**

Resolution Refutations

- **Hypotheses:**
- **$T \supset F$, $F \supset W$, W**
- **In CNF: $(\neg T \vee F) \wedge (\neg F \vee W) \wedge W$**
- **Conclusion: T**
- **Conjunction of Clauses for Resolution Refutation:**
- **$(\neg T \vee F) \wedge (\neg F \vee W) \wedge W \wedge \neg T$**

- **Resolution on F:**
- **$(\neg T \vee W) \wedge W \wedge \neg T$**

- **Simplification:**
- **$W \wedge \neg T$**

- **No further resolution is possible.**

- **Therefore, the argument is invalid and the conclusion that Jim went to a pub on Thursday is incorrect.**

- **(He could have been forced to work during the weekend for other reasons.)**

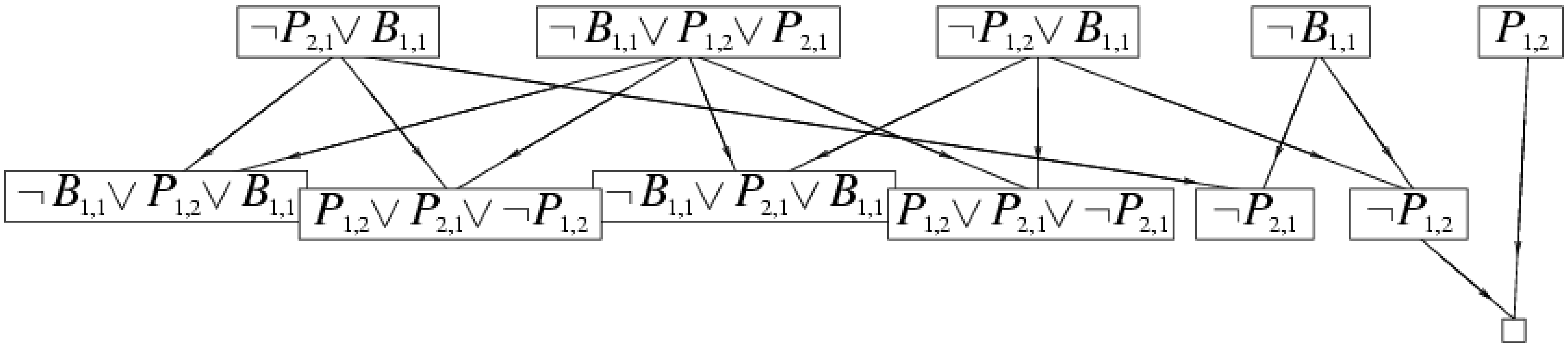
Resolution algorithm

- Proof by contradiction, i.e., show $KB \wedge \neg \alpha$ unsatisfiable

```
function PL-RESOLUTION(KB, a)returns true or false
   clauses ← the set of clauses in the CNF representation of KB ∧ ¬a
   new ←{ }
   loop do
      for each Ci, Cj in clauses do
         resolvents ← PL-RESOLVE(Ci, Cj)
         if resolvents contains the empty clause then return true
          new←new ∪ resolvents
      if new⊆ clauses then return false
      clauses←clauses∪new
```

Resolution example

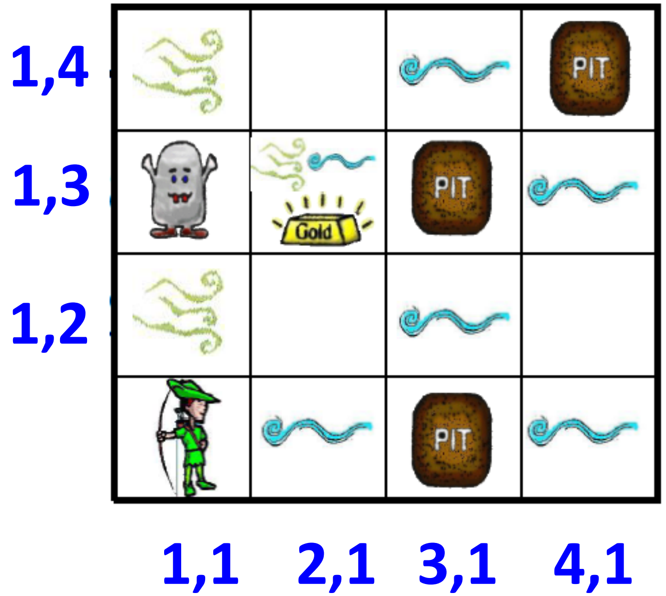
- $KB = (B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})) \wedge \neg B_{1,1}$
- $\alpha = \neg P_{1,2}$
- $(\neg P_{2,1} \vee B_{1,1}) \wedge (\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge (\neg P_{1,2} \vee B_{1,1})$



If a set of clauses is unsatisfiable, then the resolution closure of those clausescontains the empty clause.

Logical agent in the Wumpus world

- Start
  - $R_1{:}\neg P_{1,1}$
- Pits cause breezes in adjacent squares
  - $R_2{:}B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$
  - $R_3{:}B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$
- Perceiving
  - $R_4{:}\neg B_{1,1}$
    - $R_5{:}B_{2,1}$
- KB $\models \neg P_{1,2}$?
- CNF representation of KB ...
- resolution ...



{PVQ,¬PVQ,PV¬Q,¬PV¬Q} is unsatisfiable

1) PVQ	
2) ¬PVQ	
3) PV¬Q	
4) ¬PV¬Q	
5) Q	1,2
6) P	1,3
7) PV¬P	1,4
8) QV¬Q	1,4
9) QV¬Q	2,3
10) PV¬P	2,3
11) ¬P	2,4
12) ¬Q	3,4

14) PVQ	1,8
15) PVQ	1,9
16) PVQ	1,10
17) Q	1,11
18) P	1,12
19) Q	2,6
20) ¬PVQ	2,7
21) ¬PVQ	2,8
22) ¬PVQ	2,9
23) ¬PVQ	2,10
24) ¬P	2,12
25) P	3,5
26) PV¬Q	3,7

27) PV¬Q	3,8
28) PV¬Q	3,9
29) PV¬Q	3,10
30) ¬Q	3,11
31) ¬P	4,5
32) ¬Q	4,6
33) ¬PV¬Q	4,7
34) ¬PV¬Q	4,8
35) ¬PV¬Q	4,9
36) ¬PV¬Q	4,10
37) Q	5,8
38) Q	5,9
39) NIL	5,12

think : how to improve the efficiency?

Horn clause

Horn clause, Definite clause{7.5.3}

- One restricted form is Definite clause, which is a disjunction of literals of which exactly one is positive
  1. $(\neg P_{1,1} \vee \neg B_{1,1} \vee S_{1,1})$ is a definite clause
  2. $(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1})$ is not.
- Slightly more general is the Horn clause, which is a disjunction of literals of which at most one is positive.
- All definite clauses are Horn clauses, as are clauses with on positive literals, these are called goal clauses.
- Horn clauses are closed under resolution: if you resolve two Horn clauses, you get back a Horn clause.

Horn clause, Definite clause{7.5.3}

- Every definite clause can be written as an implication whose premise is a conjunction of positive literals and whose conclusion is a single positive literal.

  $\neg L_{1,1} \vee \neg \text{Breeze} \vee B_{1,1}$ can be expressed as

  $L_{1,1} \wedge \text{Breeze} \Rightarrow B_{1,1}$
- Inference with Horn clauses can be done through the forward-chaining and backward chaining algorithms.
- Deciding entailment with Horn clauses can be done in time that is linear in the size of the knowledge base

Forward chaining

Forward and backward chaining

- Horn Form (restricted)
  1. KB = conjunction of Horn clauses

- Horn clause : a disjunction of literals of which at most one is positive
  - E.g., $\neg L_{1,1} \vee \neg$ Breeze $\vee B_{1,1}$, $L_{1,1} \wedge$ Breeze $\Rightarrow B_{1,1}$
  - proposition symbol, (conjunction of symbols) $\Rightarrow$ symbol
  - E.g., $C \wedge (B \Rightarrow A) \wedge (C \wedge D \Rightarrow B)$,

- Modus Ponens (for Horn Form): complete for Horn KBs

$$\frac{\alpha_1, \ldots, \alpha_n, \qquad \alpha_1 \wedge \ldots \wedge \alpha_n \Rightarrow \beta}{\beta}$$

- Can be used with forward chaining or backward chaining.
- These algorithms are very natural and run in linear time

Forward chaining{7.5.4}

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←a table, where count[c] is the number of symbols in c's premise
  inferred←a table, where inferred[s] is initially false for all symbols
  agenda←a queue of symbols, initially symbols known to be true in KB

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

P□ Q
L∧M□ P
B∧L□ M
A∧P□ L
A∧B□ L
A
B

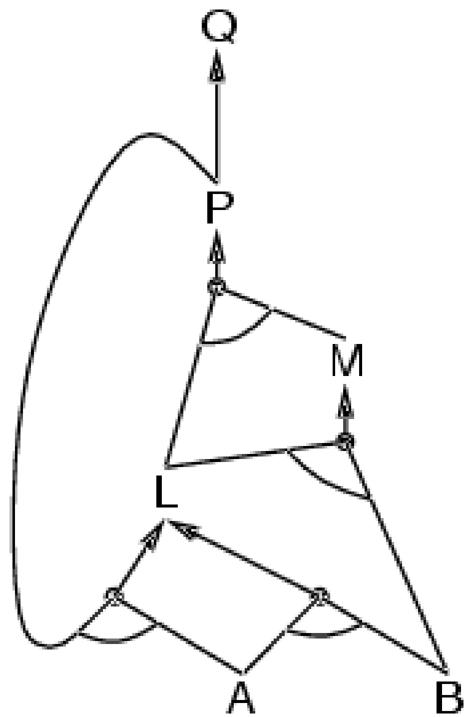
Forward chaining{7.5.4}

- The *Agenda* keeps track of symbols known to be true but not yet 'processed'.
- The *count* table keeps track of how many premises of each implication are as yet unknown. Whenever a new symbol *p* from the agenda is processed, the count is reduced by one for each implication in whose premise *p* appears. If a count reaches zero, all the premises of the implication are known, so it is conclusion can be added to the agenda.
- A symbol that is already in the set of inferred symbols need not be added to the agenda again.
- Forward chaining is an example of data-driven reasoning:
  1. reasoning in which the focus of attention starts with the known data.

Forward chaining

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

P□ Q
LΛM□ P
BΛL□ M
AΛP□ L
AΛB□ L
A
B



Forward chaining example

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

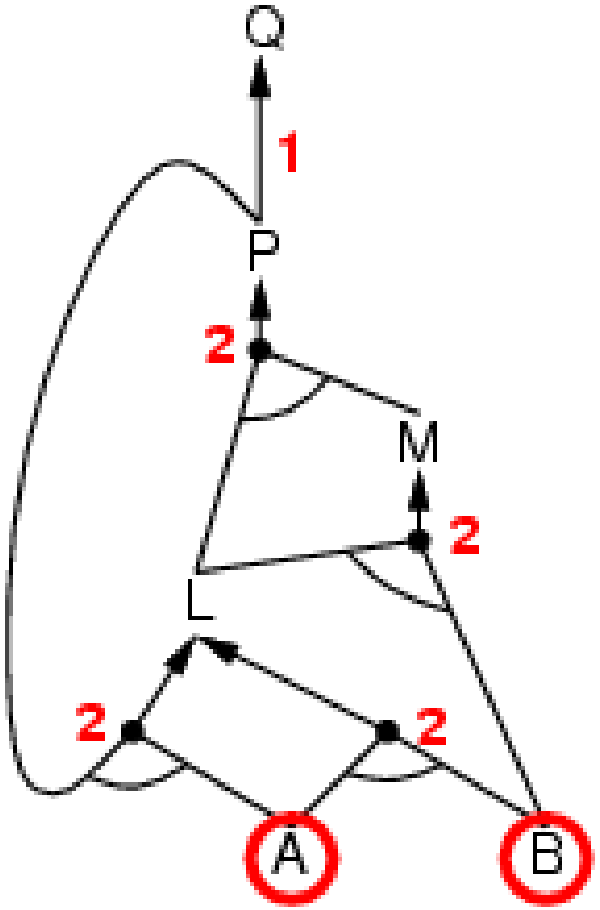
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B



A
B

Forward chaining example

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

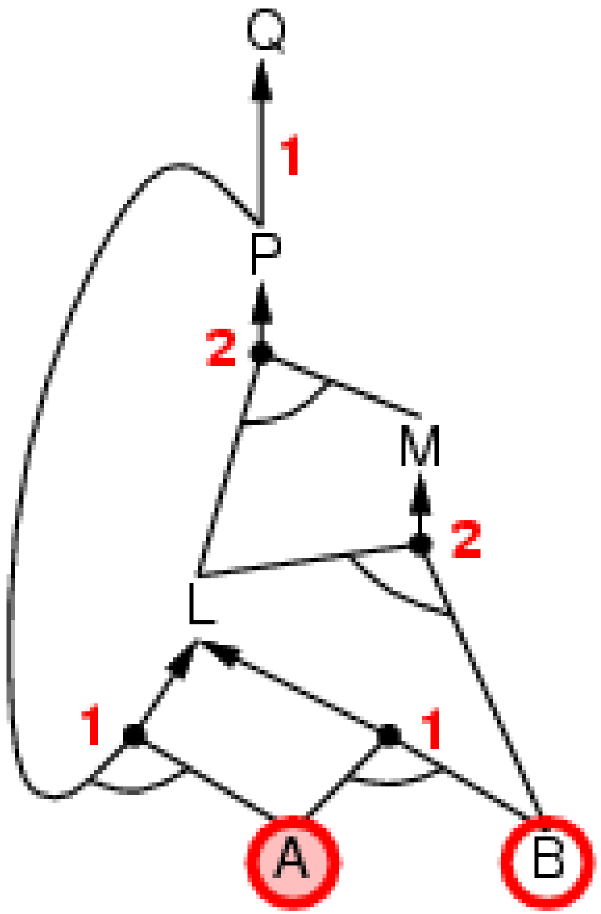
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B



$A \wedge ?P \Rightarrow ?L$

Forward chaining example

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

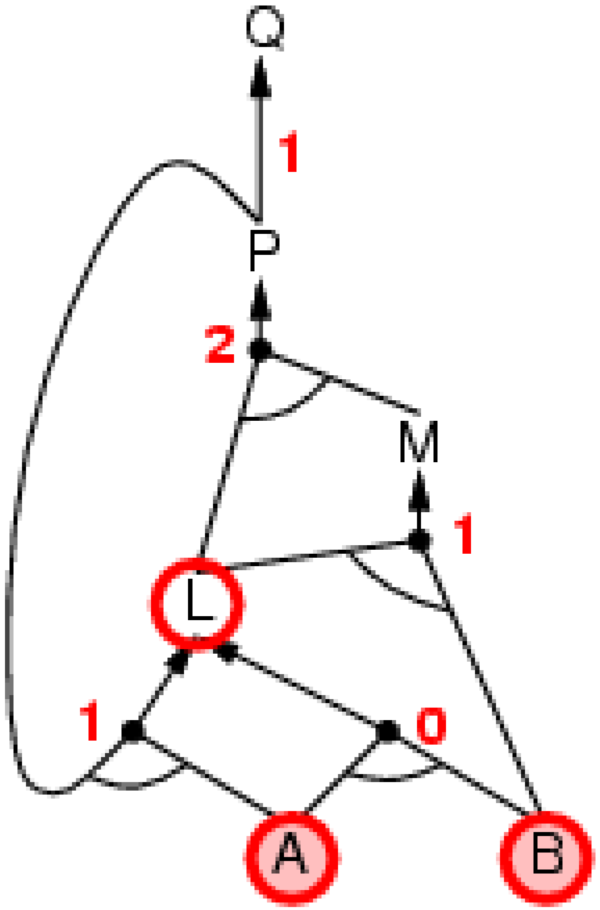
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B



$A \wedge B \Rightarrow L$

Forward chaining example

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

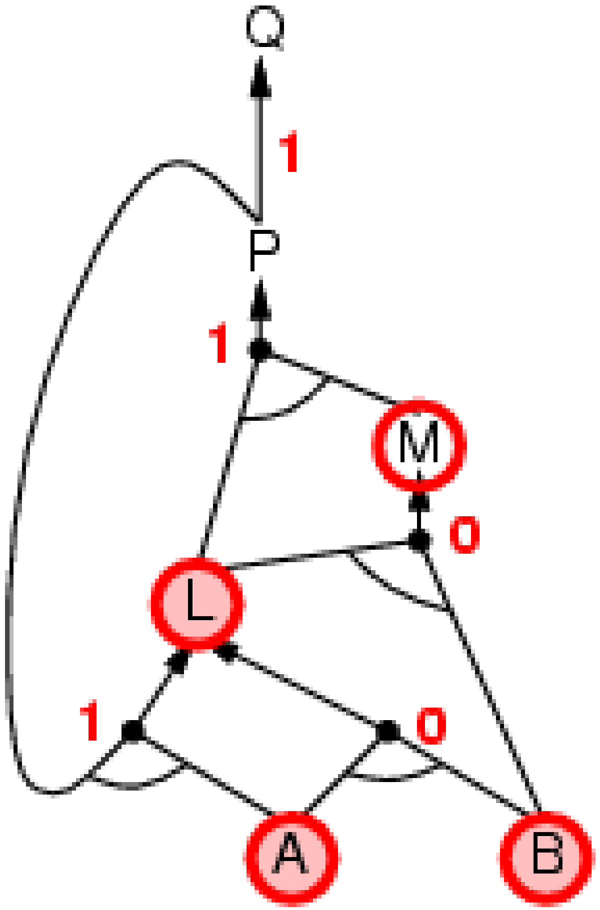
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B



$L \wedge B \Rightarrow M$

Forward chaining example

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

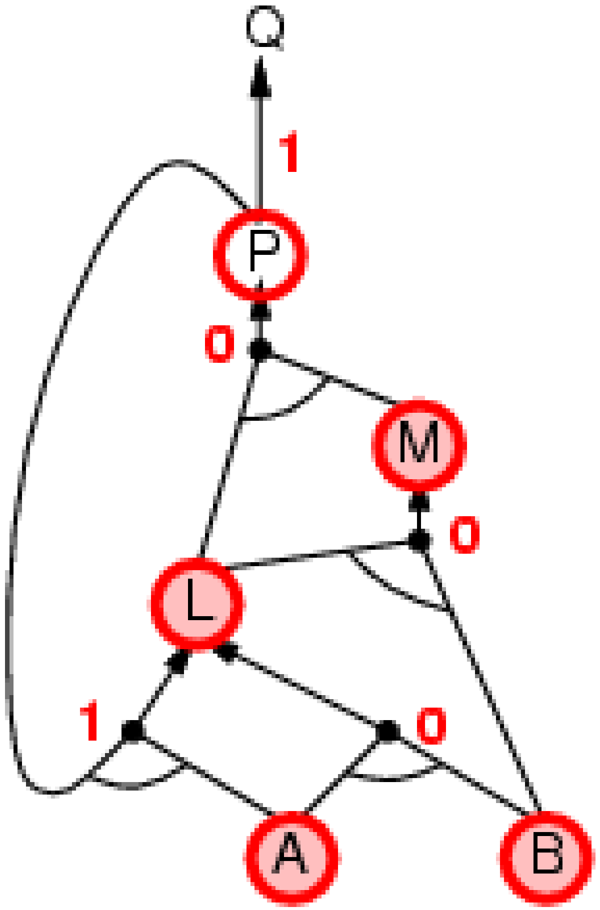
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B



$M \wedge L \Rightarrow P$

Forward chaining example

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

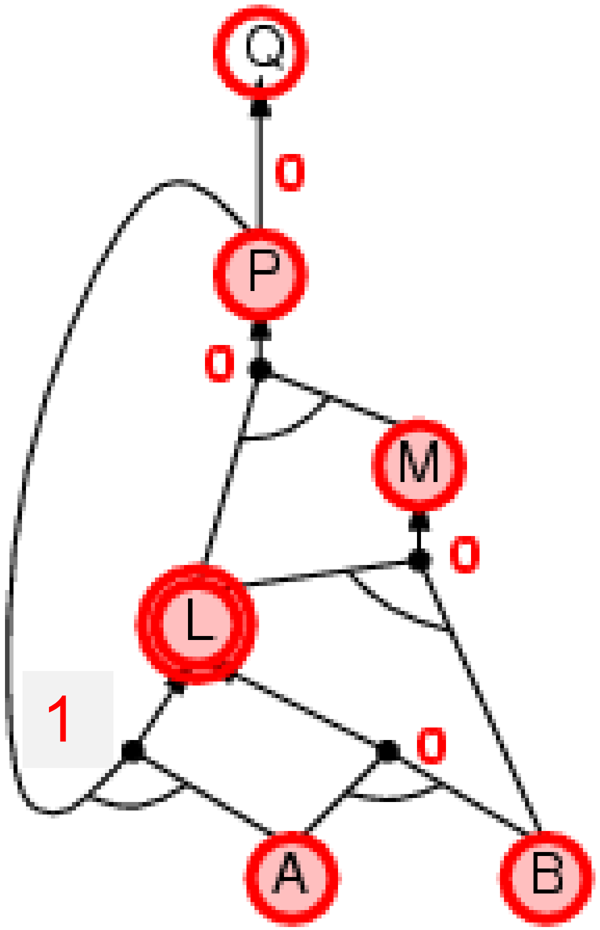
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B



$P \Rightarrow Q$

Forward chaining example

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

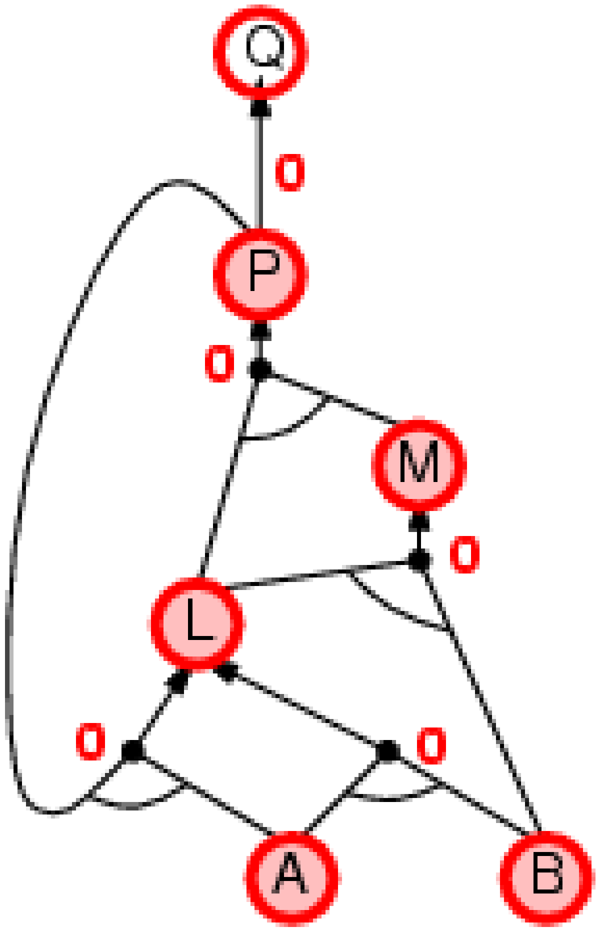
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B



$A \wedge P \Rightarrow L$

Forward chaining example

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

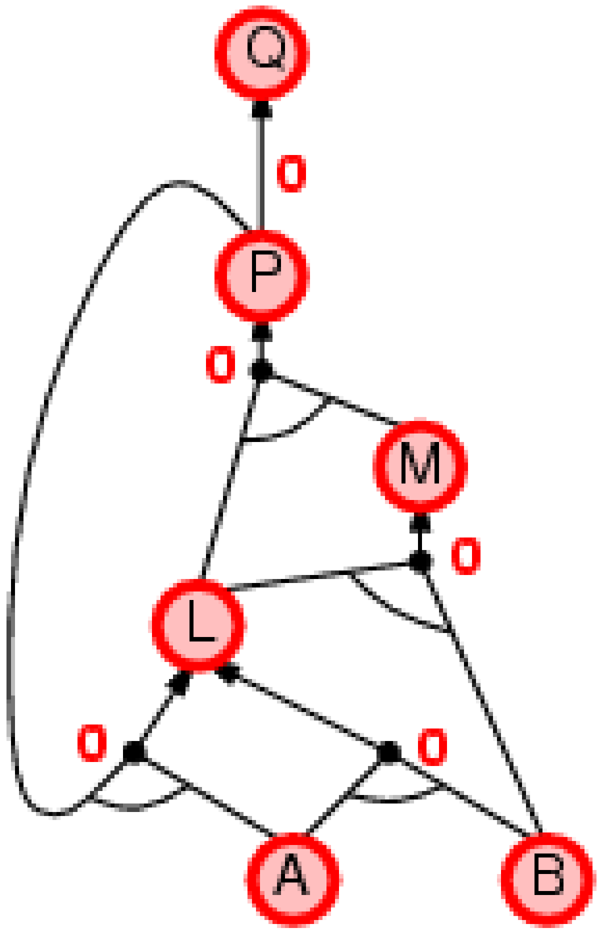
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B



$P \Rightarrow Q$

Proof of completeness

FC derives every atomic sentence that is entailed by *KB*

1. FC reaches a fixed point where no new atomic sentences are derived
2. Consider the final state as a model *m*, assigning true/false to symbols
3. Every clause in the original *KB* is true in *m*

   $$a_1 \wedge \ldots \wedge a_k \Rightarrow b$$

4. Hence *m* is a model of *KB*
5. If $KB \models q$, *q* is true in every model of *KB*, including *m*

Backward chaining

Backward chaining{7.5.4}

- Works backward from the query. If the query q is known to be true, then no work is needed. Otherwise, the algorithm finds those implications in the knowledge base whose conclusion is q. If all the premises of one of those implications can be proved true(by backward chaining), then q is true.
- As with forward chaining, an efficient implementation runs in linear time.
- Backward chaining is a form of goal-directed reasoning.

Backward chaining

Idea: work backwards from the query *q*:

to prove *q* by BC,

check if *q* is known already, or

prove by BC all premises of some rule concluding *q*

Avoid loops: check if new sub-goal is already on the goal stack

Avoid repeated work: check if new sub-goal

1. has already been proved true, or
2. has already failed

Backward chaining example

P ⇒ Q

L∧M ⇒ P

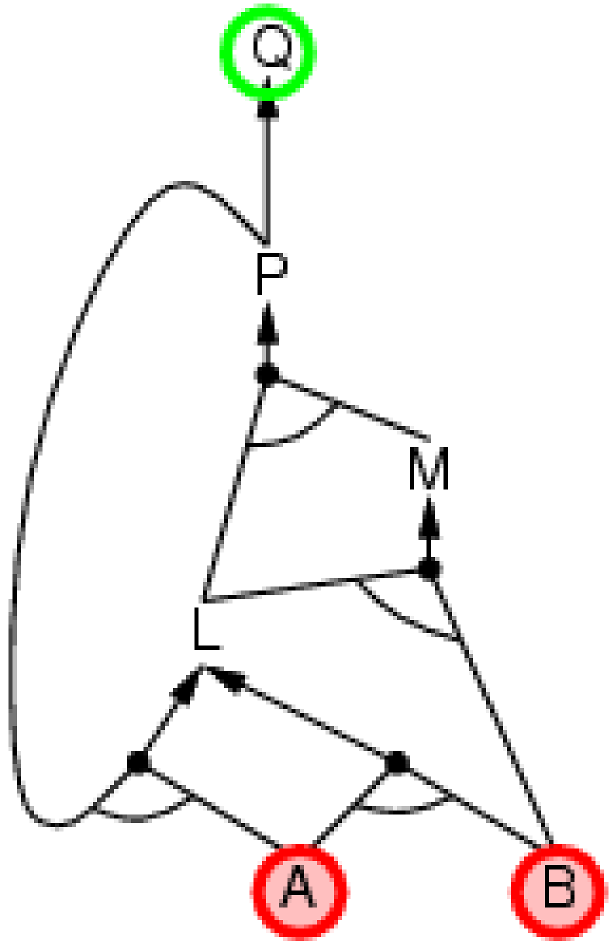
B∧L ⇒ M

A∧P ⇒ L

A∧B ⇒ L

A

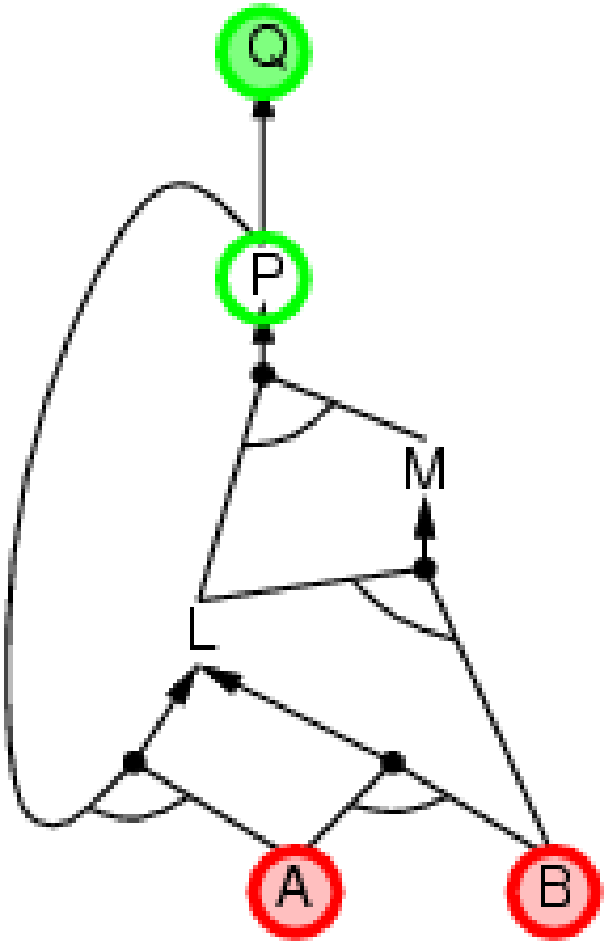
B



A
B
Q?

Backward chaining example

P ⇒ Q
L∧M ⇒ P
B∧L ⇒ M
A∧P ⇒ L
A∧B ⇒ L
A
B



A
B
Q?<=P?

Backward chaining example

P□ Q

LΛM□ P

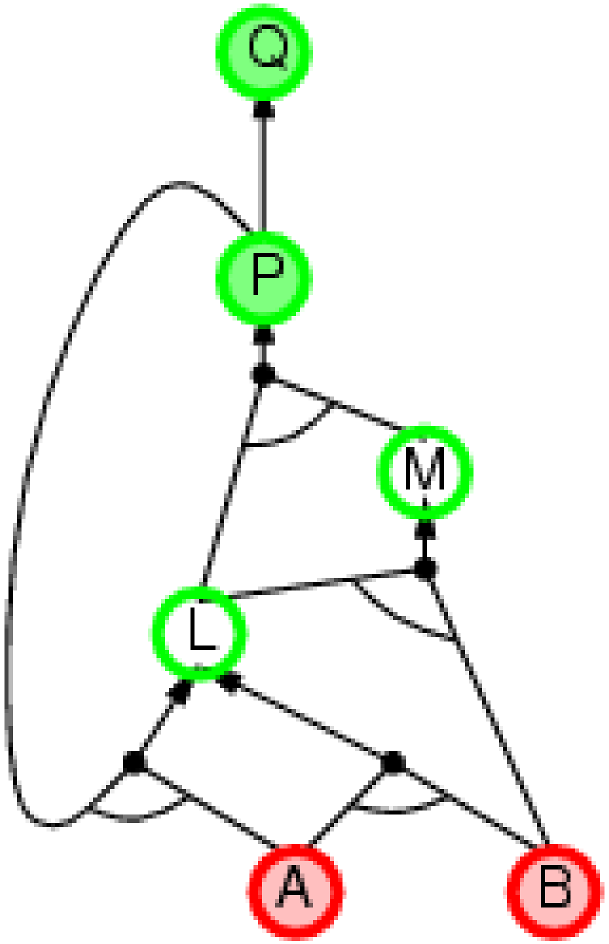
BΛL□ M

AΛP□ L

AΛB□ L

A

B



A

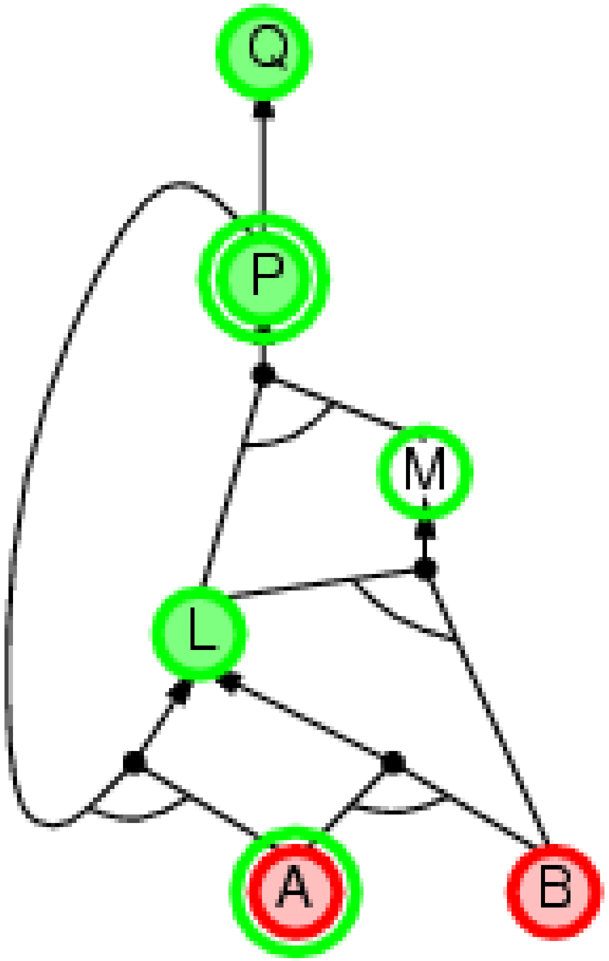
B

Q?<=P?

L? ∧ M? => P?

Backward chaining example

P□ Q
L∧M□ P
B∧L□ M
A∧P□ L
A∧B□ L
A
B



A
B
Q?<=P?
L? ∧ M? => P?
P? ∧ A => L?

Backward chaining example

P□ Q

LΛM□ P

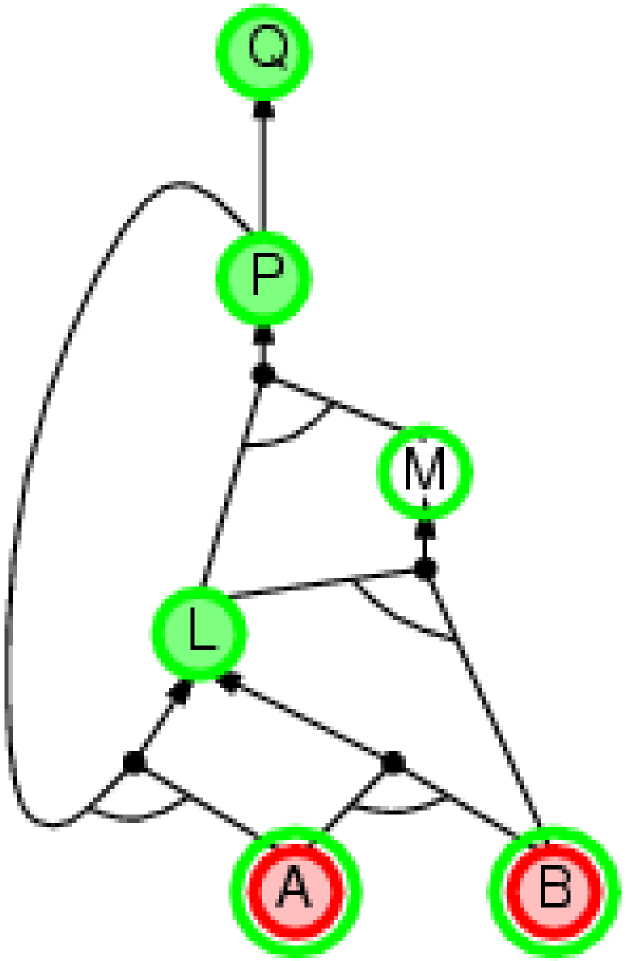
BΛL□ M

AΛP□ L

AΛB□ L

A

B



A

B

Q?<=P?

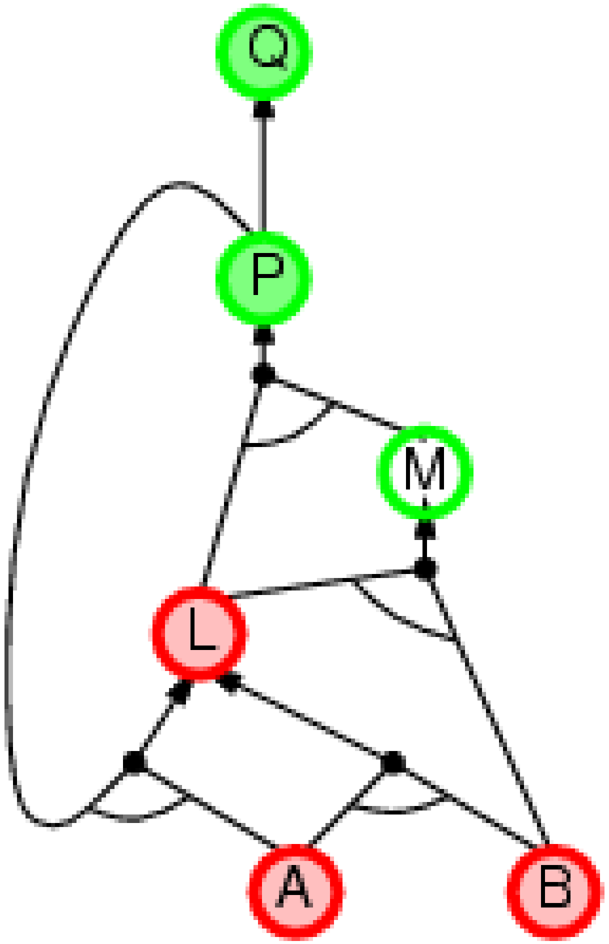
L? ∧ M? => P?

P? ∧ A => L?

A ∧ B => L

Backward chaining example

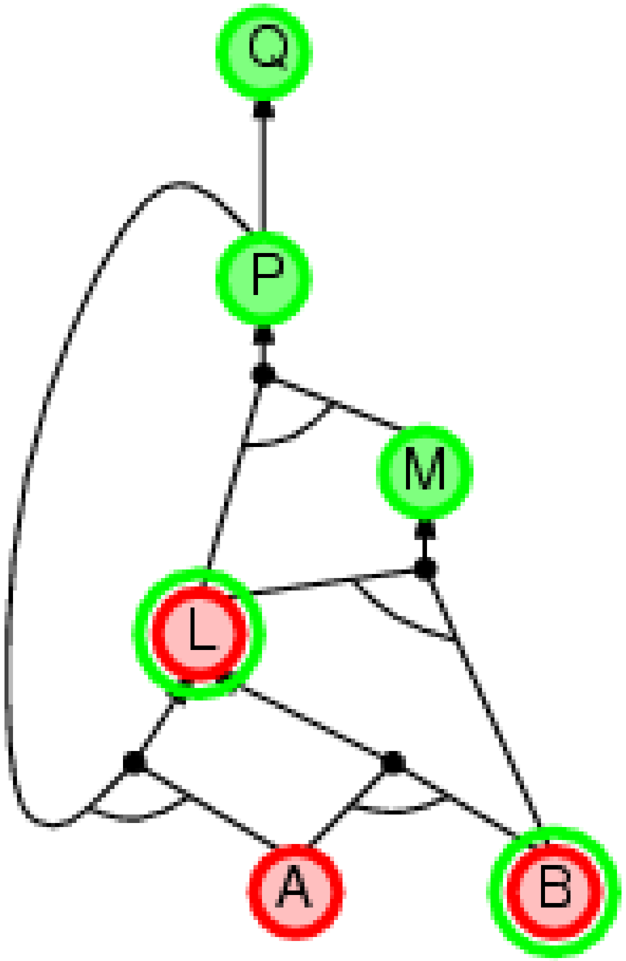
P⇒ Q
L∧M⇒ P
B∧L⇒ M
A∧P⇒ L
A∧B⇒ L
A
B



A
B
Q?<=P?
L? ∧ M? => P?
P? ∧ A => L?
A ∧ B => L

Backward chaining example

P⇒ Q
L∧M⇒ P
B∧L⇒ M
A∧P⇒ L
A∧B⇒ L
A
B



A
B
Q?<=P?
L? ∧ M? => P?
P? ∧ A => L?
A ∧ B => L
L ∧ B => M

Backward chaining example

P□ Q

LΛM□ P

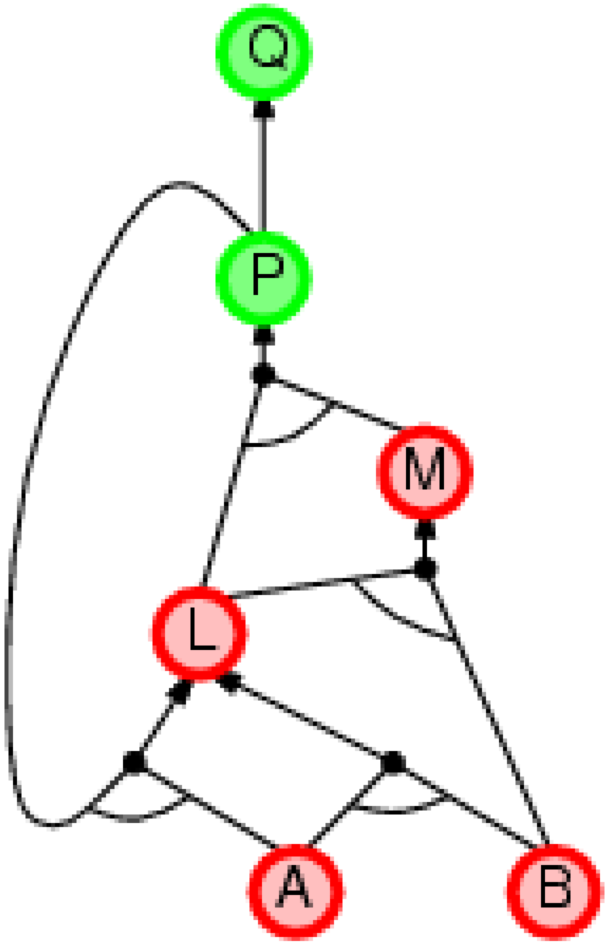
BΛL□ M

AΛP□ L

AΛB□ L

A

B



A

B

Q?<=P?

L ∧ M => P?

P? ∧ A => L?

A ∧ B => L

L ∧ B => M

Backward chaining example

P□ Q

L∧M□ P

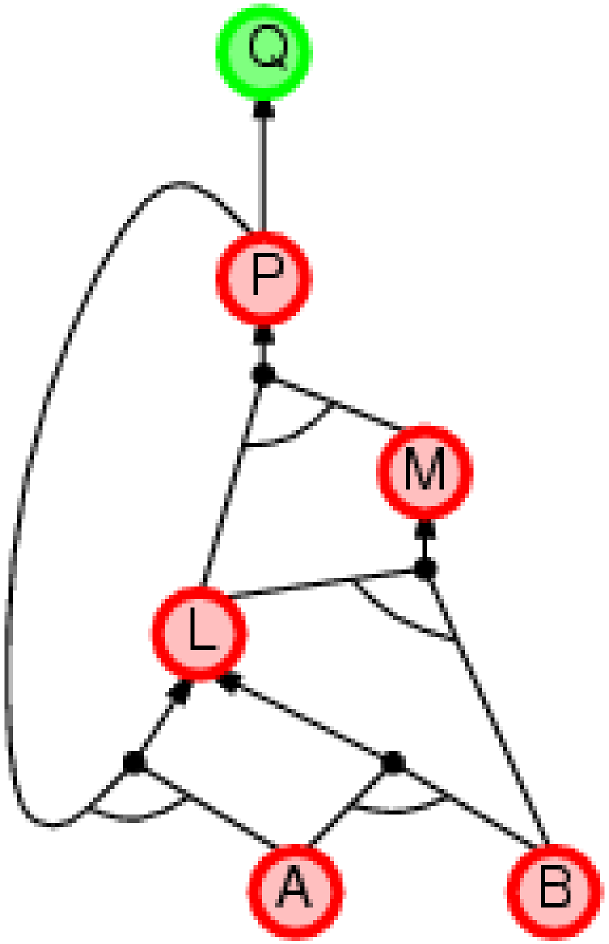
B∧L□ M

A∧P□ L

A∧B□ L

A

B



A

B

Q?<=P

L ∧ M => P

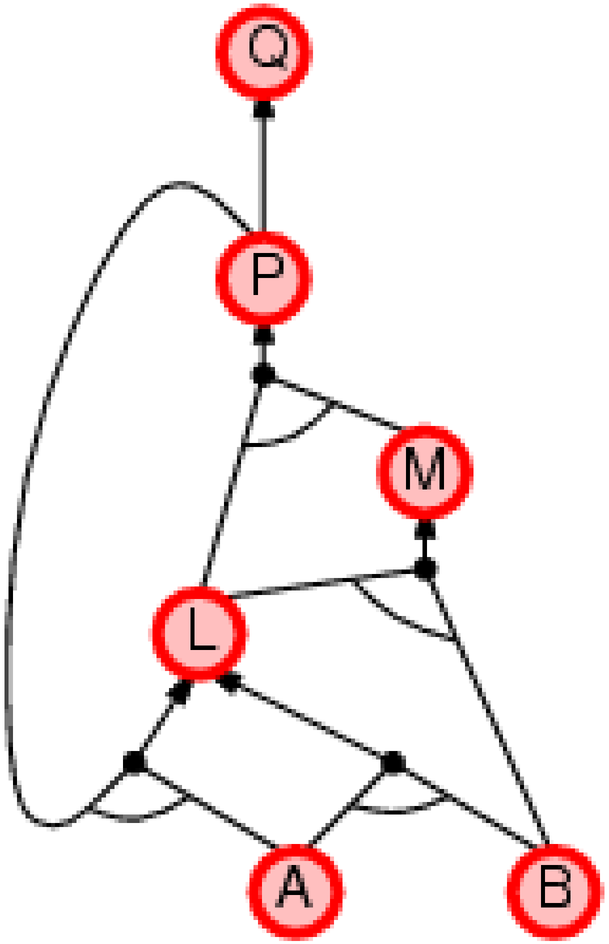
P? ∧ A => L?

A ∧ B => L

L ∧ B => M

Backward chaining example

P Q
LΛM P
BΛL M
AΛP L
AΛB L
A
B



A
B
Q<=P
L ∧ M => P
P ∧ A => L
A ∧ B => L
L ∧ B => M

Forward vs. backward chaining

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

Sumary

- Logical agents apply inference to a knowledge base to derive new information and make decisions
- Basic concepts of logic
  - syntax : formal structure of sentences
  - semantics : truth of sentences with regard to models
- Propositional logic lacks expressive power
  - can't say “ pits cause breezes in adjacent squares ”
  - except by writing one sentence for each square

Expressiveness limitation of propositional logic

- KB contains "physics" sentences for every single square

- For every time *t* and every location $[x^t, y^t]$,

  $$L^t_{x,y} \wedge FacingRight^t \wedge Forward^t \Rightarrow L^{t+1}_{x+1,y}$$

- Rapid proliferation of clauses

Summary

- Logical agents apply inference to a knowledge base to derive new information and make decisions
- Basic concepts of logic:
  1. syntax: formal structure of sentences
  2. semantics: truth of sentences wrt models
  3. entailment: necessary truth of one sentence given another
  4. inference: deriving sentences from other sentences
  5. soundness: derivations produce only entailed sentences
  6. completeness: derivations can produce all entailed sentences

Summary

- Wumpus world requires the ability to represent partial and negated information, reason by cases, etc.
- Resolution is complete for propositional logic
- Forward, backward chaining are linear-time, complete for Horn clauses
- Propositional logic lacks expressive power

Propositional Logic (PL)

- P.L. offers techniques for capturing facts or rules in symbolic form and then operates on them through use of logical operators.
- PL provide method of managing statements that are either TRUE or FALSE.
- Prolog is based on PC

Propositional Calculus

- You have seen that resolution, including resolution refutation, is a suitable tool for automated reasoning in the propositional calculus.
- If we build a machine that represents its knowledge as propositions, we can use these mechanisms to enable the machine to deduce new knowledge from existing knowledge and verify hypotheses about the world.
- However, propositional calculus has some serious restrictions in its capability to represent knowledge.

Propositional Calculus

- In propositional calculus, atoms have no internal structure; we cannot reuse the same proposition for a different object, but each proposition always refers to the same object.
- For example, in the toy block world, the propositions ON_A_B and ON_A_C are completely different from each other.
- We could as well call them PETER and BOB instead.
- So if we want to express rules that apply to a whole class of objects, in propositional calculus we would have to define separate rules for every single object of that class.

Predicate Calculus

- So it is a better idea to use predicates instead of propositions.

- This leads us to predicate calculus in next lecture.

- Predicate calculus has symbols called
  - object constants,
  - relation constants, and
  - function constants

- These symbols will be used to refer to objects in the world and to propositions about the word.

Thank you

End of Chapter 7-2