CS 1571 Introduction to AI
Lecture 13

Propositional logic

Milos Hauskrecht
milos@cs.pitt.edu
5329 Sennott Square

Logical inference problem

Logical inference problem:
- **Given:**
  - a knowledge base KB (a set of sentences) and
  - a sentence α (called **a theorem)**,
- How is the logical inference problem defined?

Logical inference problem

Logical inference problem:

- **Given:**
  - a knowledge base KB (a set of sentences) and
  - a sentence α (called **a theorem)**,
- **Does a KB semantically entail** α ? $KB \models \alpha$?

In other words: In all interpretations in which sentences in the KB are true, is also α true?

Approaches:

- **?**

Logical inference problem

Logical inference problem:

- **Given:**
  - a knowledge base KB (a set of sentences) and
  - a sentence α (called **a theorem)**,
- **Does a KB semantically entail** α ? $KB \models \alpha$?

In other words: In all interpretations in which sentences in the KB are true, is also α true?

Approaches:

- **Truth-table approach**
- **Inference rules**
- **Conversion to SAT**
  - **Resolution refutation**

Inference problem and satisfiability

How is the logical inference problem related to the satisfiability problem?

Inference problem and satisfiability

How is the logical inference problem related to the satisfiability problem?

$KB \models \alpha$ if and only if

$(KB \wedge \neg \alpha)$ is **unsatisfiable**

Universal inference rule: Resolution rule

Sometimes inference rules can be combined into a single rule

Resolution rule

- sound inference rule that works for KB in CNF
- It is complete for **propositional logic (refutation complete)**

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

A	B	C	$A \vee B$	$\neg B \vee C$	$A \vee C$
False	*False*	*False*	*False*	*True*	*False*
False	*False*	*True*	*False*	*True*	*True*
False	*True*	*False*	*True*	*False*	*False*
False	*True*	*True*	*True*	*True*	*True*
True	*False*	*False*	*True*	*True*	*True*
True	*False*	*True*	*True*	*True*	*True*
True	*True*	*False*	*True*	*False*	*True*
True	*True*	*True*	*True*	*True*	*True*

Resolution algorithm

Algorithm:

- **Convert KB to the CNF form;**
- **Apply iteratively the resolution rule** starting from
  $KB, \neg \alpha$ (in the CNF form)
- **Stop when**:
  - Contradiction (empty clause) is reached:
    - $A, \neg A \rightarrow \emptyset$
    - proves entailment.
  - No more new sentences can be derived
    - disproves it.

Example. Resolution.

KB: $(P \wedge Q) \wedge (P \Rightarrow R) \wedge [(Q \wedge R) \Rightarrow S]$ **Theorem:** S

Step 1. convert KB to CNF:

- $P \wedge Q \longrightarrow P \wedge Q$
- $P \Rightarrow R \longrightarrow (\neg P \vee R)$
- $(Q \wedge R) \Rightarrow S \longrightarrow (\neg Q \vee \neg R \vee S)$

KB: $P \quad Q \quad (\neg P \vee R) \quad (\neg Q \vee \neg R \vee S)$

Step 2. Negate the theorem to prove it via refutation

$S \longrightarrow \neg S$

Step 3. Run resolution on the set of clauses

$P \quad Q \quad (\neg P \vee R) \quad (\neg Q \vee \neg R \vee S) \quad \neg S$

Example. Resolution.

KB: $(P \wedge Q) \wedge (P \Rightarrow R) \wedge [(Q \wedge R) \Rightarrow S]$ **Theorem:** S

$P \quad Q \quad (\neg P \vee R) \quad (\neg Q \vee \neg R \vee S) \quad \neg S$

Example. Resolution.

KB: $(P \wedge Q) \wedge (P \Rightarrow R) \wedge [(Q \wedge R) \Rightarrow S]$ **Theorem:** S

$$P \quad Q \quad (\neg P \vee R) \quad (\neg Q \vee \neg R \vee S) \quad \neg S$$

$$R$$

Example. Resolution.

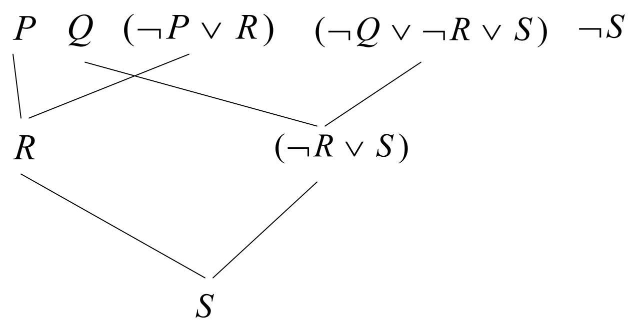
KB: $(P \wedge Q) \wedge (P \Rightarrow R) \wedge [(Q \wedge R) \Rightarrow S]$ **Theorem:** S

$$P \quad Q \quad (\neg P \vee R) \quad (\neg Q \vee \neg R \vee S) \quad \neg S$$

$$R \qquad (\neg R \vee S)$$

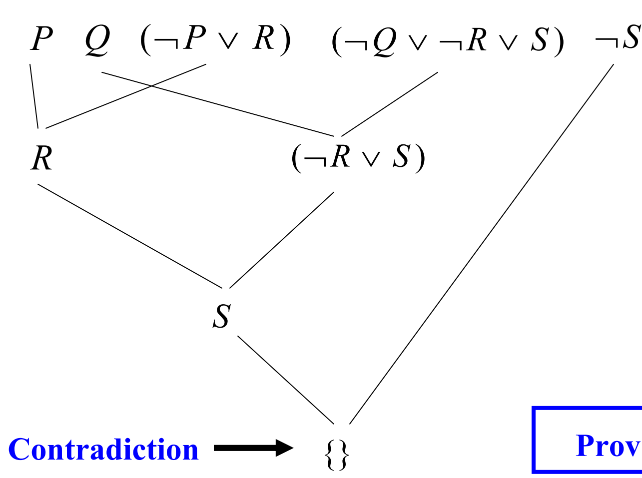
Example. Resolution.

KB: $(P \wedge Q) \wedge (P \Rightarrow R) \wedge [(Q \wedge R) \Rightarrow S]$ **Theorem:** S



Example. Resolution.

KB: $(P \wedge Q) \wedge (P \Rightarrow R) \wedge [(Q \wedge R) \Rightarrow S]$ **Theorem:** S



Proved: S

KB in restricted forms

If the sentences in the KB are restricted to some special forms some of the sound inference rules may become complete

Example:

- **Horn form (Horn normal form)**

$$(A \vee \neg B) \wedge (\neg A \vee \neg C \vee D)$$

Can be written also as:

$$(B \Rightarrow A) \wedge ((A \wedge C) \Rightarrow D)$$

Resolution (or modus ponens) are sound and complete for inferences on propositional symbols in the HNF

KB in Horn form

- **Horn form:** a clause with **at most one positive literal**

$$(A \vee \neg B) \wedge (\neg A \vee \neg C \vee D)$$

- **Note: Not all sentences in propositional logic can be converted into the Horn form**
- **KB in Horn normal form**:
  - Two types of propositional statements:
    - **Rules** $(\neg B_1 \vee \neg B_2 \vee \dots \neg B_k \vee A)$

$$(\neg (B_1 \wedge B_2 \wedge \dots B_k) \vee A)$$

$$(B_1 \wedge B_2 \wedge \dots B_k \Rightarrow A)$$

$\equiv$

    - Propositional symbols: **facts** B

KB in Horn form

- **Application of the resolution rule:**
  - Infers new facts from previous facts

$$\frac{(A \vee \neg B), B}{A} \qquad \frac{(A \vee \neg B), \quad (B \vee \neg C)}{(A \vee C)}$$

  - Resolution is **sound and complete** for inferences on propositional symbols for KB in the Horn normal form (clausal form)

- Similarly, **modus ponens is sound and complete** when the HNF is written in the implicative form

Complexity of inferences for KBs in HNF

Question: How efficient are the inferences in the HNF?

- **If we consider only inferences on propositional symbols**
- **procedures linear in the size of the KB in the HNF exist.**

Terminology:

- Size of a clause: the number of literals it contains.
- Size of the KB in the HNF: the sum of the sizes of its elements.

Example:

$$A, B, (A \wedge B \Rightarrow C), (C \Rightarrow D), (C \Rightarrow E), (E \wedge F \Rightarrow G)$$

or

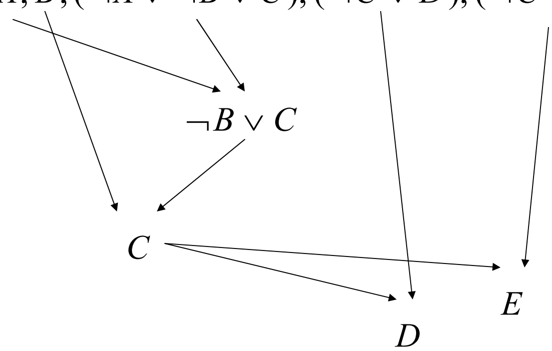
$$A, B, (\neg A \vee \neg B \vee C), (\neg C \vee D), (\neg C \vee E), (\neg E \vee \neg F \vee G)$$

The size is: 12

Complexity of inferences for KBs in HNF

How to do the inference? If the HNF (is in the clausal form) we can apply resolution.

$$A, B, (\neg A \vee \neg B \vee C), (\neg C \vee D), (\neg C \vee E), (\neg E \vee \neg F \vee G)$$

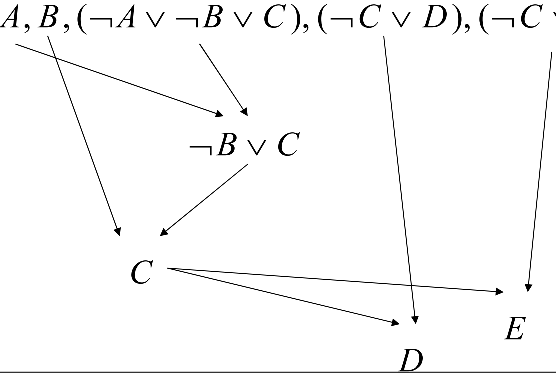


Complexity of inferences for KBs in HNF

Features:

- Every resolution is a **positive unit resolution**; that is, a resolution in which **one clause is a positive unit clause** (i.e., a proposition symbol).

$$A, B, (\neg A \vee \neg B \vee C), (\neg C \vee D), (\neg C \vee E), (\neg E \vee \neg F \vee G)$$

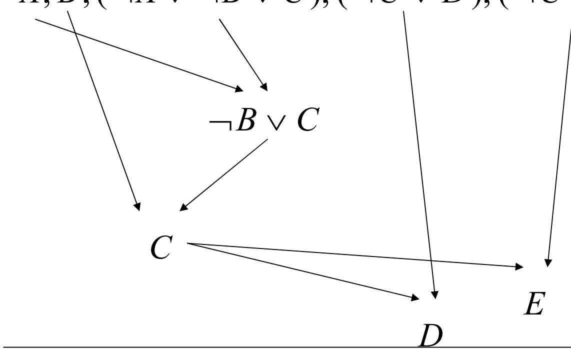


Complexity of inferences for KBs in HNF

Features:

- At each resolution, the input clause which is not a unit clause is a logical consequence of the result of the resolution. (Thus, the input clause may be deleted upon completion of the resolution operation.)

$$A, B, (\neg A \vee \neg B \vee C), (\neg C \vee D), (\neg C \vee E), (\neg E \vee \neg F \vee G)$$

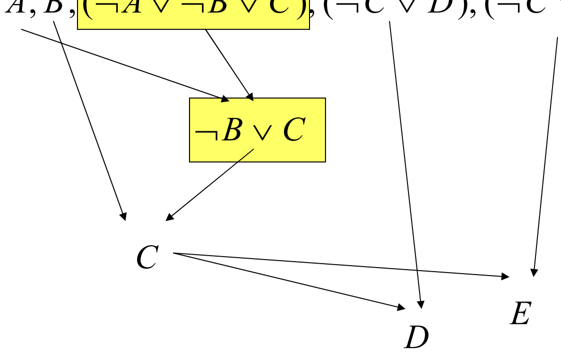


Complexity of inferences for KBs in HNF

Features:

- At each resolution, the input clause which is not a unit clause is a logical consequence of the result of the resolution. (Thus, the input clause may be deleted upon completion of the resolution operation.)

$$A, B, (\neg A \vee \neg B \vee C), (\neg C \vee D), (\neg C \vee E), (\neg E \vee \neg F \vee G)$$

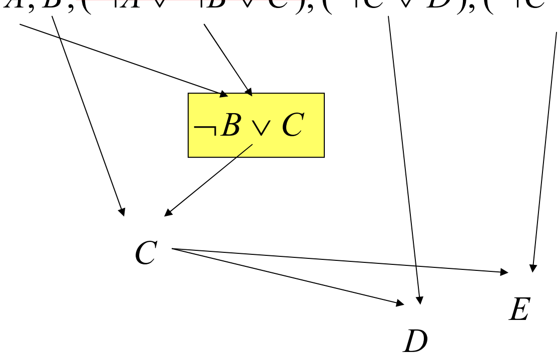


Complexity of inferences for KBs in HNF

Features:

- Following this deletion, the size of the KB (the sum of the lengths of the remaining clauses) is one less than it was before the operation.)

$A, B,$ ~~$(\neg A \vee \neg B \vee C)$~~, $(\neg C \vee D), (\neg C \vee E), (\neg E \vee \neg F \vee G)$

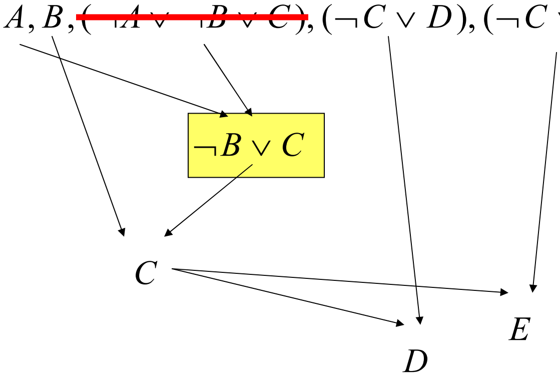


Complexity of inferences for KBs in HNF

Features:

- If n is the size of the KB, then at most n positive unit resolutions may be performed on it.

$A, B,$ ~~$(\neg A \vee \neg B \vee C)$~~, $(\neg C \vee D), (\neg C \vee E), (\neg E \vee \neg F \vee G)$



Complexity of inferences for KBs in HNF

A linear time resolution algorithm:

- **The number of positive unit resolutions is limited to the size of the formula (n)**
- **But to assure overall linear time we need to access each proposition in a constant time:**
- Data structures indexed by proposition names may be accessed in constant time. (This is possible if the proposition names are number in a range (e.g., 1..n), so that array lookup is the access operation.
- If propositions are accessed by name, then a symbol table is necessary, and the algorithm will run in time $O(n \cdot \log(n))$.

Forward and backward chaining

Two inference procedures based on **modus ponens** for **Horn KBs**:

- **Forward chaining**

  Idea: Whenever the premises of a rule are satisfied, infer the conclusion. Continue with rules that became satisfied.

- **Backward chaining (goal reduction)**

  Idea: To prove the fact that appears in the conclusion of a rule prove the premises of the rule. Continue recursively.

Both procedures are **complete for KBs in the Horn form** !!!

Forward chaining example

- **Forward chaining**

  Idea: Whenever the premises of a rule are satisfied, infer the conclusion. Continue with rules that became satisfied.

Assume the KB with the following rules and facts:

KB: R1: $A \wedge B \Rightarrow C$

R2: $C \wedge D \Rightarrow E$

R3: $C \wedge F \Rightarrow G$

F1: A

F2: B

F3: D

Theorem: E ?

Forward chaining example

Theorem: E

KB: R1: $A \wedge B \Rightarrow C$

R2: $C \wedge D \Rightarrow E$

R3: $C \wedge F \Rightarrow G$

F1: A

F2: B

F3: D

Forward chaining example

Theorem: E

KB: R1: $A \wedge B \Rightarrow C$

R2: $C \wedge D \Rightarrow E$

R3: $C \wedge F \Rightarrow G$

F1: A

F2: B

F3: D

Rule R1 is satisfied.

F4: C

Forward chaining example

Theorem: E

KB: R1: $A \wedge B \Rightarrow C$

R2: $C \wedge D \Rightarrow E$

R3: $C \wedge F \Rightarrow G$

F1: A

F2: B

F3: D

Rule R1 is satisfied.

F4: C

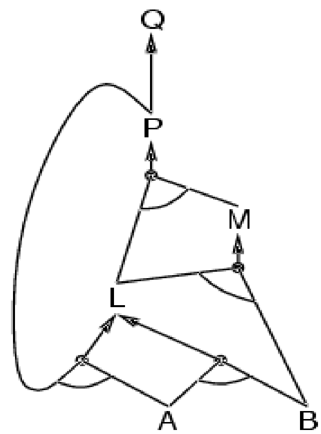
Rule R2 is satisfied.

F5: E ✓

Forward chaining

- Efficient implementation: linear in the size of the KB
- **Example:**

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



Forward chaining

- Runs in time linear in the number of literals in the Horn formulae

```
function PL-FC-ENTAILS?(KB, q) returns true or false
  local variables: count, a table, indexed by clause, initially the number of premises
                   inferred, a table, indexed by symbol, each entry initially false
                   agenda, a list of symbols, initially the symbols known to be true

  while agenda is not empty do
      p ← POP(agenda)
      unless inferred[p] do
          inferred[p] ← true
          for each Horn clause c in whose premise p appears do
              decrement count[c]
              if count[c] = 0 then do
                  if HEAD[c] = q then return true
                  PUSH(HEAD[c], agenda)
  return false
```

Forward chaining

**

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

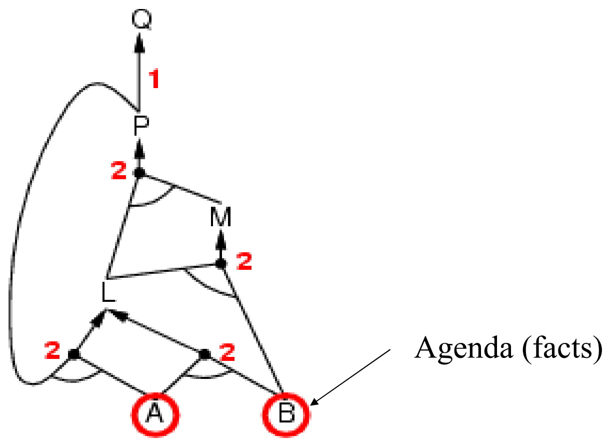
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B



Forward chaining

•

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

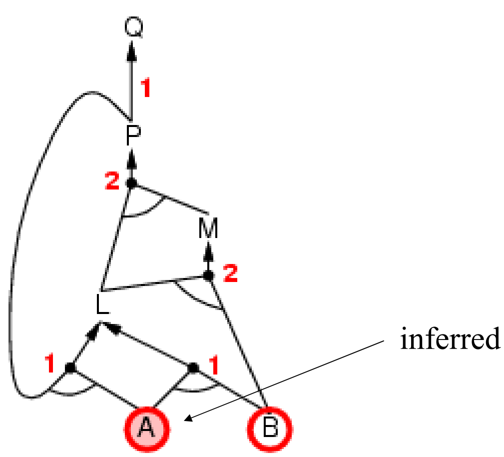
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B



Forward chaining

•

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

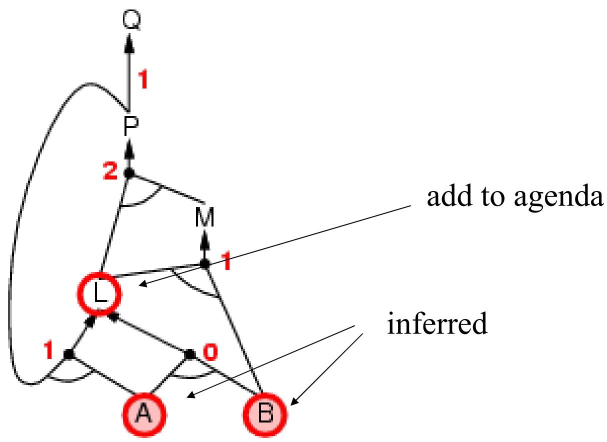
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B



Forward chaining

•

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

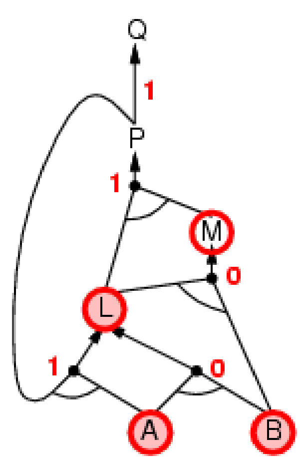
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B



Forward chaining

•

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

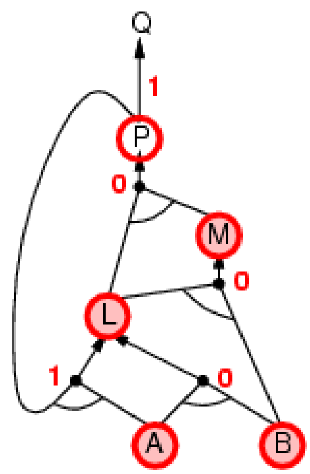
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B



Forward chaining

•

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

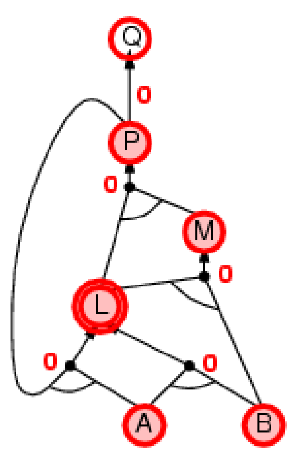
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B



Forward chaining

•

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

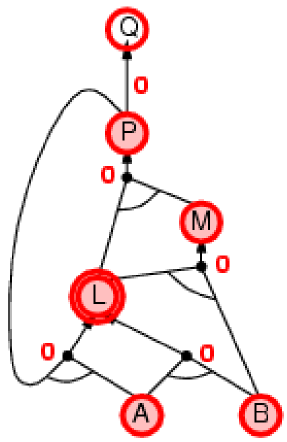
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

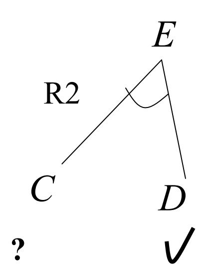
$A \wedge B \Rightarrow L$

A

B



Backward chaining example



KB: R1: $A \wedge B \Rightarrow C$

R2: $C \wedge D \Rightarrow E$

R3: $C \wedge F \Rightarrow G$

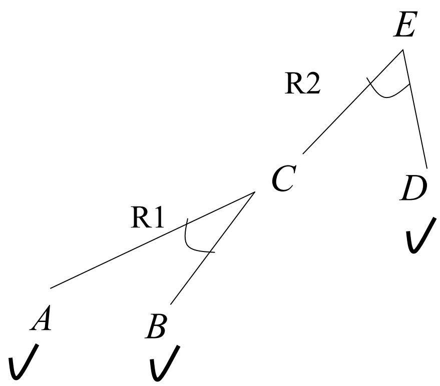
F1: A

F2: B

F3: D

- Backward chaining is more focused:
  - tries to prove the theorem only

Backward chaining example



KB: R1: $A \wedge B \Rightarrow C$

R2: $C \wedge D \Rightarrow E$

R3: $C \wedge F \Rightarrow G$

F1: A

F2: B

F3: D

- Backward chaining is more focused:
  - tries to prove the theorem only

Backward chaining

•

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

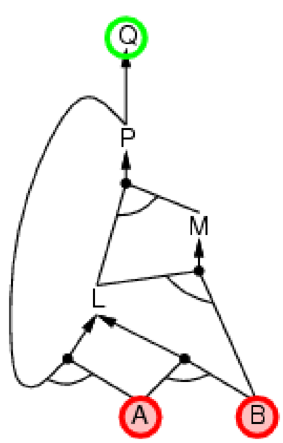
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B



Backward chaining

•

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

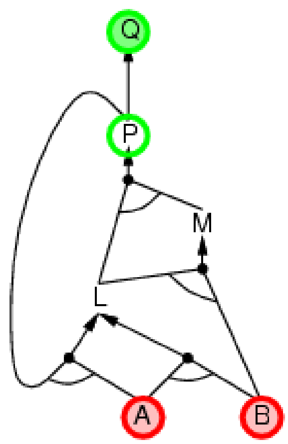
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B



Backward chaining

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

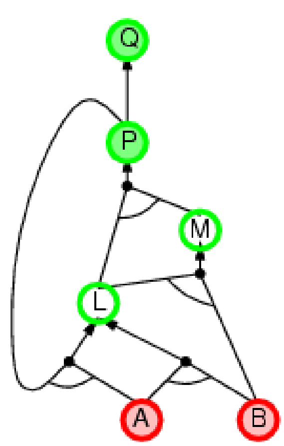
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B



Backward chaining

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

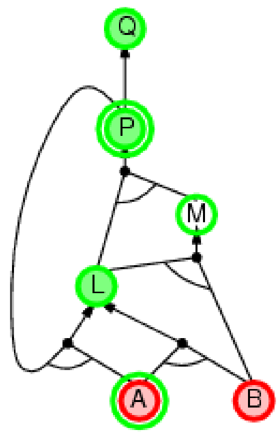
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B



Backward chaining

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

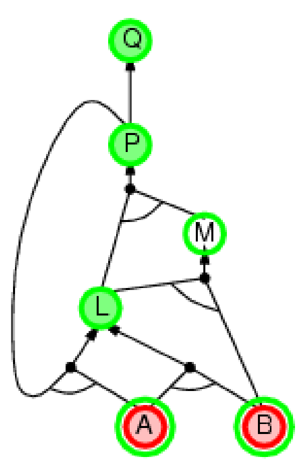
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B



Backward chaining

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

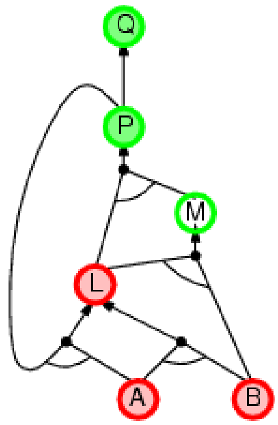
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B



Backward chaining

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

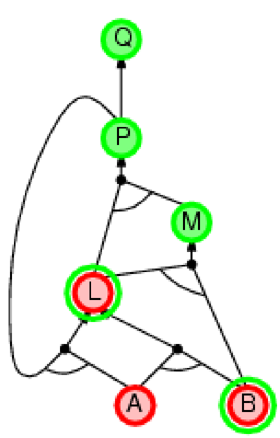
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B



Backward chaining

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

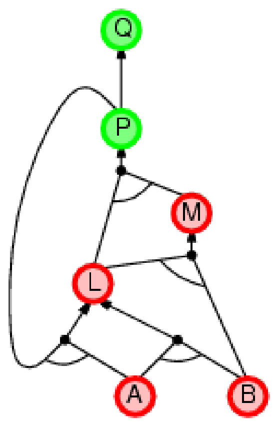
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B



Backward chaining

•

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

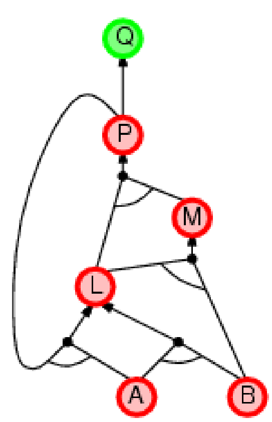
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B



Backward chaining

•

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

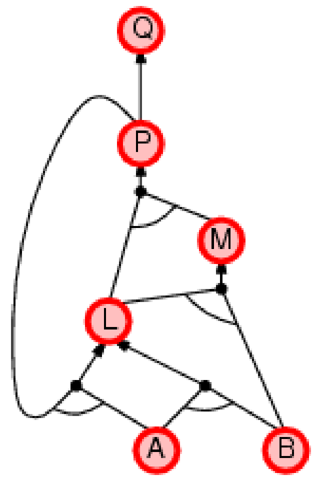
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B



Forward vs Backward chaining

- **FC is data-driven**, automatic, unconscious processing,
  - 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,
  - 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**

KB agents based on propositional logic

- Propositional logic allows us to build **knowledge-based agents** capable of answering queries about the world by inferring new facts from the known ones
- **Example:** an agent for diagnosis of a bacterial disease

Facts: The stain of the organism is gram-positive
The growth conformation of the organism is chains

Rules: **(If)** The stain of the organism is gram-positive $\wedge$
The morphology of the organism is coccus $\wedge$
The growth conformation of the organism is chains
(Then) $\Rightarrow$ The identity of the organism is streptococcus

First order logic

Limitations of propositional logic

The world we want to represent and reason about consists of a number of objects with variety of properties and relations among them

Propositional logic:

- Represents statements about the world without reflecting this structure and without modeling these entities explicitly

Consequence:

- some knowledge is hard or impossible to encode in the propositional logic.
- Two cases that are hard to represent:
  - **Statements about similar objects, relations**
  - **Statements referring to groups of objects**.

Limitations of propositional logic

- **Statements about similar objects and relations needs to be enumerated**
- **Example:** Seniority of people domain

Assume we have: *John is older than Mary*
Mary is older than Paul

To derive *John is older than Paul* we need:

John is older than Mary $\wedge$ *Mary is older than Paul*
$\Rightarrow$ *John is older than Paul*

Assume we add another fact: *Jane is older than Mary*

To derive *Jane is older than Paul* we need:

Jane is older than Mary $\wedge$ *Mary is older than Paul*
$\Rightarrow$ *Jane is older than Paul*

What is the problem?

Limitations of propositional logic

- **Statements about similar objects and relations needs to be enumerated**
- **Example:** Seniority of people domain

Assume we have: *John is older than Mary*
Mary is older than Paul

To derive *John is older than Paul* we need:

John is older than Mary $\wedge$ *Mary is older than Paul*
$\Rightarrow$ *John is older than Paul*

Assume we add another fact: *Jane is older than Mary*

To derive *Jane is older than Paul* we need:

Jane is older than Mary $\wedge$ *Mary is older than Paul*
$\Rightarrow$ *Jane is older than Paul*

Problem: KB grows large

Limitations of propositional logic

- **Statements about similar objects and relations needs to be enumerated**
- **Example:** Seniority of people domain

For inferences we need:

John is older than Mary $\wedge$ *Mary is older than Paul*
$\Rightarrow$ *John is older than Paul*

Jane is older than Mary $\wedge$ *Mary is older than Paul*
$\Rightarrow$ *Jane is older than Paul*

- **Problem:** if we have many people and facts about their seniority we need represent many rules like this to allow inferences
- **Possible solution: ??**

Limitations of propositional logic

- **Statements about similar objects and relations needs to be enumerated**
- **Example:** Seniority of people domain

  For inferences we need:

  John is older than Mary $\wedge$ *Mary is older than Paul*
  $\Rightarrow$ *John is older than Paul*

  Jane is older than Mary $\wedge$ *Mary is older than Paul*
  $\Rightarrow$ *Jane is older than Paul*

- **Problem:** if we have many people and facts about their seniority we need represent many rules like this to allow inferences
- **Possible solution: introduce variables**

  PersA *is older than* ***PersB*** $\wedge$ ***PersB*** *is older than* ***PersC***
  $\Rightarrow$ ***PersA*** *is older than* ***PersC***

Limitations of propositional logic

- **Statements referring to groups of objects require exhaustive enumeration of objects**
- **Example:**

  Assume we want to express *Every student likes vacation*

  Doing this in propositional logic would require to include statements about every student

  John likes vacation $\wedge$
  Mary likes vacation $\wedge$
  Ann likes vacation $\wedge$
  …

- **Solution:** Allow quantification in statements

First-order logic (FOL)

- More expressive than **propositional logic**

- **Eliminates deficiencies of PL by:**
  - Representing objects, their properties, relations and statements about them;
  - Introducing variables that refer to an arbitrary objects and can be substituted by a specific object
  - Introducing quantifiers allowing us to make statements over groups objects without the need to represent each of them separately