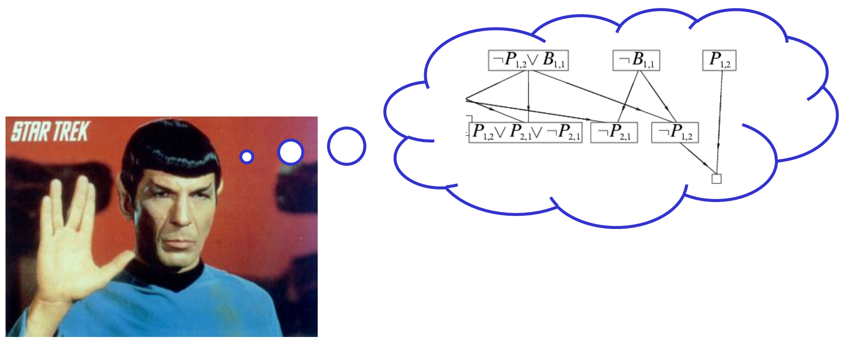
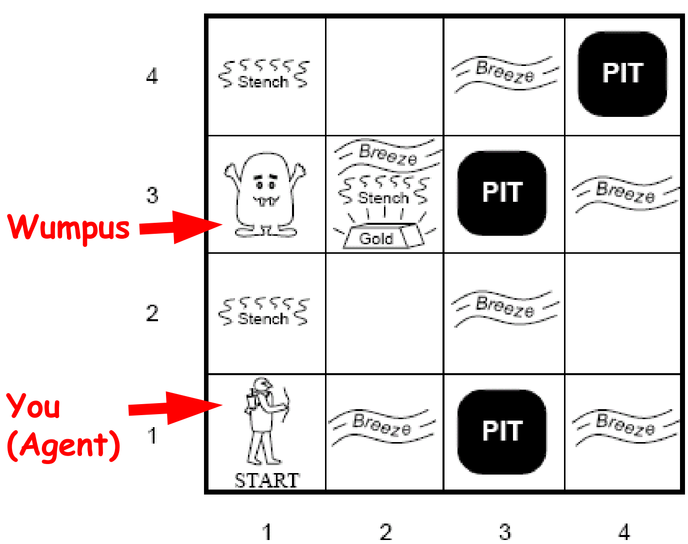
CSE 473

Chapter 7

Inference Techniques for Logical Reasoning



Recall: Wumpus World



Recall: Wumpus KB

Knowledge Base (KB) includes the following sentences:

- Statements currently known to be true:

  $\neg P_{1,1}$
  $\neg B_{1,1}$
  $B_{2,1}$

- Properties of the world: E.g., "Pits cause breezes in adjacent squares"

  $B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$
  $B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$
  (and so on for all squares)

1,4	2,4	3,4	4,4
1,3	2,3	3,3	4,3
1,2 OK	2,2 P?	3,2	4,2
1,1 V OK	2,1 $\boxed{A}$ B OK	3,1 P?	4,1

Is there no pit in [1,2]?

KB $\models \neg P_{1,2}$?

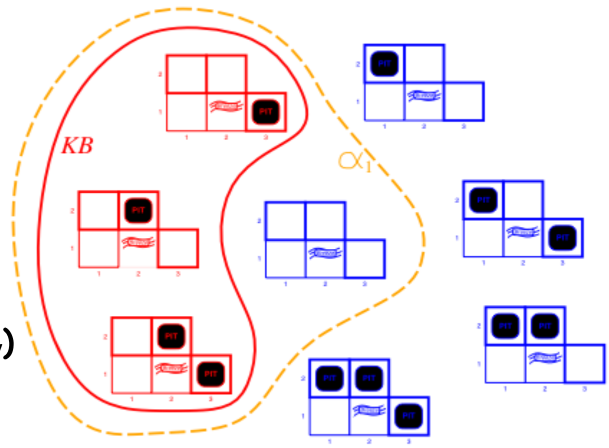
Recall from last time:

m **is a model of a sentence α if** **α is true in *m***

M(α) **is the set of <u>all models</u> of α**

KB $\models \alpha$ (KB "entails" α) iff *M(KB)* $\subseteq$ *M(α)*

$\alpha_1 = \neg P_{1,2}$



M(KB) $\subseteq$ ***M(α_1)***

KB = wumpus-world rules + observations

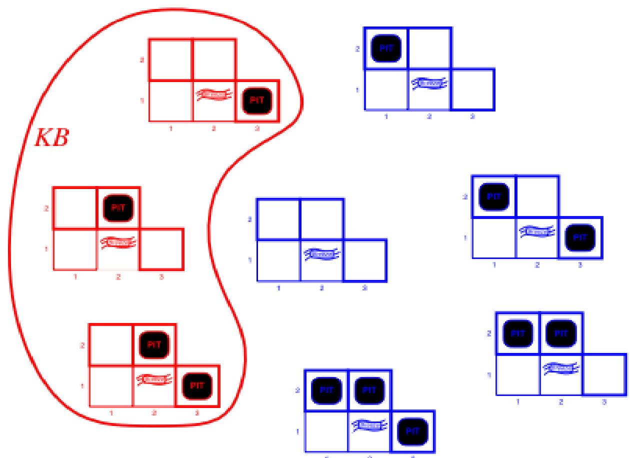
Inference by Truth Table Enumeration

$B_{1,1}$	$B_{2,1}$	$P_{1,1}$	$P_{1,2}$	$P_{2,1}$	$P_{2,2}$	$P_{3,1}$	KB	$\neg P_{1,2}$
false	*false*	*false*	*false*	*false*	*false*	*false*	*false*	*true*
false	*false*	*false*	*false*	*false*	*false*	*true*	*false*	*true*
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
false	*true*	*false*	*false*	*false*	*false*	*false*	*false*	*true*
false	*true*	*false*	*false*	*false*	*false*	*true*	*true*	*true*
false	*true*	*false*	*false*	*false*	*true*	*false*	*true*	*true*
false	*true*	*false*	*false*	*false*	*true*	*true*	*true*	*true*
false	*true*	*false*	*false*	*true*	*false*	*false*	*false*	*true*
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
true	*true*	*true*	*true*	*true*	*true*	*true*	*false*	*false*

In all models in which KB is true, $\neg P_{1,2}$ is also true
Therefore, KB $\models \neg P_{1,2}$

Another Example

Is there a pit in [2,2]?



7

Inference by Truth Table Enumeration

$B_{1,1}$	$B_{2,1}$	$P_{1,1}$	$P_{1,2}$	$P_{2,1}$	$P_{2,2}$	$P_{3,1}$	KB
false	*false*	*false*	*false*	*false*	*false*	*false*	*false*
false	*false*	*false*	*false*	*false*	*false*	*true*	*false*
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
false	*true*	*false*	*false*	*false*	*false*	*false*	*false*
false	*true*	*false*	*false*	*false*	*false*	*true*	*true*
false	*true*	*false*	*false*	*false*	*true*	*false*	*true*
false	*true*	*false*	*false*	*false*	*true*	*true*	*true*
false	*true*	*false*	*false*	*true*	*false*	*false*	*false*
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
true	*true*	*true*	*true*	*true*	*true*	*true*	*false*

$P_{2,2}$ is false in a model in which KB is true
Therefore, KB $\not\models P_{2,2}$

8

Inference by TT Enumeration

- Algorithm: Depth-first enumeration of all models (see Fig. 7.10 in text for pseudocode)
- - Algorithm is sound & complete

- For n symbols:
- time complexity = $O(2^n)$, space = $O(n)$

9

Concepts for Other Techniques: Logical Equivalence

Two sentences are logically equivalent iff they are true in the same models: $\alpha \equiv \beta$ iff $\alpha \models \beta$ and $\beta \models \alpha$

$$
\begin{aligned}
(\alpha \wedge \beta) &\equiv (\beta \wedge \alpha) \quad \text{commutativity of } \wedge \\
(\alpha \vee \beta) &\equiv (\beta \vee \alpha) \quad \text{commutativity of } \vee \\
((\alpha \wedge \beta) \wedge \gamma) &\equiv (\alpha \wedge (\beta \wedge \gamma)) \quad \text{associativity of } \wedge \\
((\alpha \vee \beta) \vee \gamma) &\equiv (\alpha \vee (\beta \vee \gamma)) \quad \text{associativity of } \vee \\
\neg(\neg\alpha) &\equiv \alpha \quad \text{double-negation elimination} \\
(\alpha \Rightarrow \beta) &\equiv (\neg\beta \Rightarrow \neg\alpha) \quad \text{contraposition} \\
(\alpha \Rightarrow \beta) &\equiv (\neg\alpha \vee \beta) \quad \text{implication elimination} \\
(\alpha \Leftrightarrow \beta) &\equiv ((\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)) \quad \text{biconditional elimination} \\
\neg(\alpha \wedge \beta) &\equiv (\neg\alpha \vee \neg\beta) \quad \text{de Morgan} \\
\neg(\alpha \vee \beta) &\equiv (\neg\alpha \wedge \neg\beta) \quad \text{de Morgan} \\
(\alpha \wedge (\beta \vee \gamma)) &\equiv ((\alpha \wedge \beta) \vee (\alpha \wedge \gamma)) \quad \text{distributivity of } \wedge \text{ over } \vee \\
(\alpha \vee (\beta \wedge \gamma)) &\equiv ((\alpha \vee \beta) \wedge (\alpha \vee \gamma)) \quad \text{distributivity of } \vee \text{ over } \wedge
\end{aligned}
$$

10

Concepts for Other Techniques: Validity and Satisfiability

- A sentence is *valid* if it is true in *all* models (a tautology)
  e.g., *True*, $A \vee \neg A$, $A \Rightarrow A$, $(A \wedge (A \Rightarrow B)) \Rightarrow B$
- Validity is connected to inference via the Deduction Theorem:
  $KB \models \alpha$ if and only if $(KB \Rightarrow \alpha)$ is valid
- A sentence is *satisfiable* if it is true in *some* model
  e.g., $A \vee B$, C
- A sentence is *unsatisfiable* if it is true in no models
  e.g., $A \wedge \neg A$
- Satisfiability is connected to inference via the following: $KB \models \alpha$ if and only if $(KB \wedge \neg\alpha)$ is unsatisfiable (proof by contradiction)

11

Inference/Proof Techniques

- Two kinds (roughly):

  Model checking
  - Truth table enumeration (always exponential in *n*)
  - Efficient backtracking algorithms,
    e.g., Davis-Putnam-Logemann-Loveland (DPLL)
  - Local search algorithms (sound but incomplete)
    e.g., randomized hill-climbing (WalkSAT)

  Successive application of inference rules
  - Generate new sentences from old in a sound way
  - Proof = a sequence of inference rule applications
  - Use inference rules as *successor function* in a standard search algorithm

12

Inference Technique I: Resolution

Terminology:

Literal = proposition symbol or its negation

E.g., A, $\neg A$, B, $\neg B$, etc.

Clause = disjunction of literals

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

Resolution assumes sentences are in

Conjunctive Normal Form (CNF):

sentence = conjunction of clauses

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

Conversion to CNF

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

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

   This is in CNF – Done!

Resolution motivation

There is a pit in [1,3] or
There is a pit in [2,2]

There is no pit in [2,2]

There is a pit in [1,3]

More generally,

$$\frac{l_1 \vee \ldots \vee l_k, \qquad \neg l_i}{l_1 \vee \ldots \vee l_{i-1} \vee l_{i+1} \vee \ldots \vee l_k}$$

15

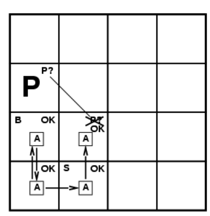
Inference Technique: Resolution

- General 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., $$\frac{P_{1,3} \vee P_{2,2}, \qquad \neg P_{2,2}}{P_{1,3}}$$



- Resolution is sound for propositional logic

16

Resolution

Soundness of resolution inference rule
(Recall logical equivalence $A \Rightarrow B \equiv \neg A \lor B$)

$$\neg(l_1 \lor \ldots \lor l_{i-1} \lor l_{i+1} \lor \ldots \lor l_k) \Rightarrow l_i$$
$$\neg m_j \Rightarrow (m_1 \lor \ldots \lor m_{j-1} \lor m_{j+1} \lor \ldots \lor m_n)$$

$$\neg(l_i \lor \ldots \lor l_{i-1} \lor l_{i+1} \lor \ldots \lor l_k) \Rightarrow (m_1 \lor \ldots \lor m_{j-1} \lor m_{j+1} \lor \ldots \lor m_n)$$

(since $l_i = \neg m_j$)

17

Resolution algorithm

- To show **KB** $\models \alpha$, use proof by contradiction, i.e., show $KB \land \neg\alpha$ unsatisfiable

```
function PL-RESOLUTION(KB, α) returns true or false
   clauses ← the set of clauses in the CNF representation of KB ∧ ¬α
   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
```

18

Resolution example

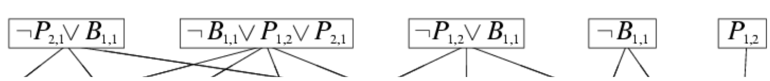
Given no breeze in [1,1], prove there's no pit in [1,2]

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

Resolution: Convert to CNF and show KB $\wedge \neg \alpha$ is unsatisfiable

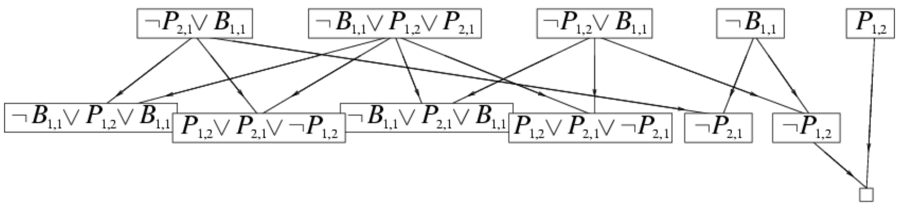
19

Resolution example



20

Resolution example



Empty clause
(i.e., KB $\wedge \neg \alpha$ unsatisfiable)

21

Inference Technique II: Forward/Backward Chaining

- Require sentences to be in **Horn Form**:
  - KB = conjunction of Horn clauses
  - Horn clause =
    - proposition symbol or
    - "(conjunction of symbols) $\Rightarrow$ symbol"
    - (i.e. clause with at most 1 positive literal)
  - E.g., KB = $C \wedge (B \Rightarrow A) \wedge (C \wedge D \Rightarrow B)$
- F/B chaining based on "Modus Ponens" rule:

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

  Complete for Horn clauses
- Very natural and linear time complexity in size of KB

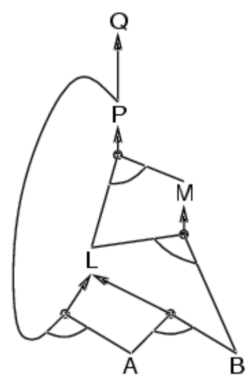
22

Forward chaining

- Idea: fire any rule whose premises are satisfied in *KB*, add its conclusion to *KB*, until query q is found

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

Query = "Is Q true?"



AND-OR Graph

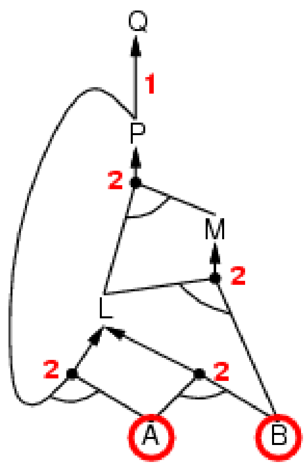
Forward chaining algorithm

```
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
```

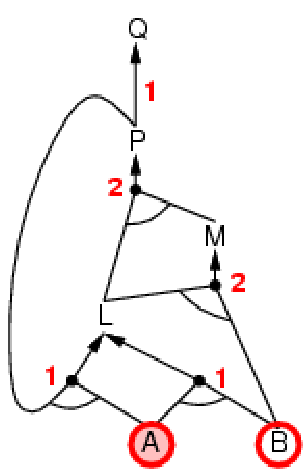
Forward chaining is sound & complete for Horn KB

Forward chaining example

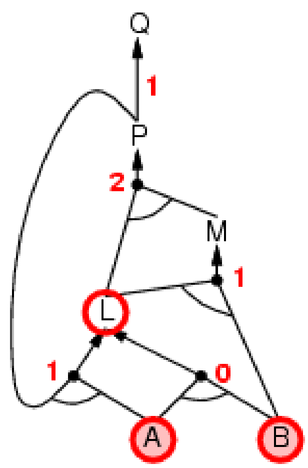


Query = Q
(i.e. "Is Q true?")

Forward chaining example

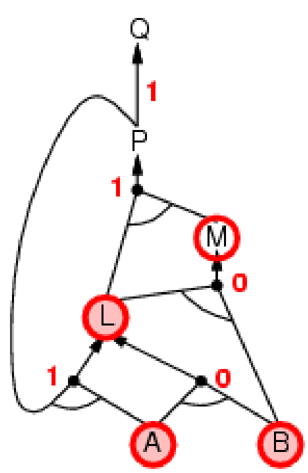


Forward chaining example



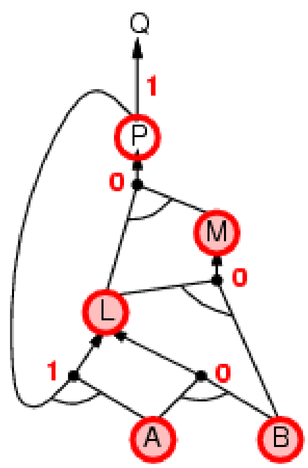
27

Forward chaining example

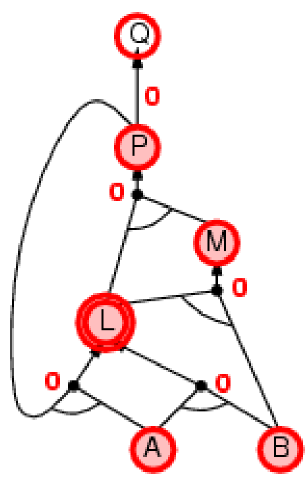


28

Forward chaining example



Forward chaining example



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

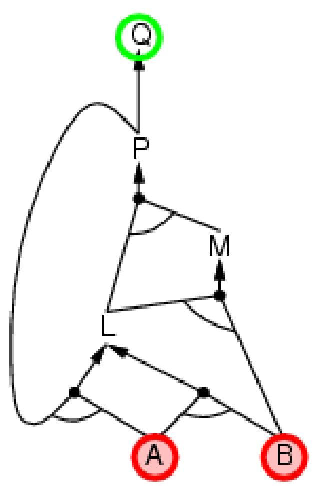
Avoid loops: check if new subgoal is already on goal stack

Avoid repeated work: check if new subgoal

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

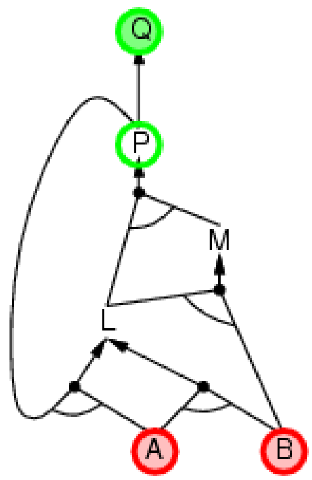
31

Backward chaining example



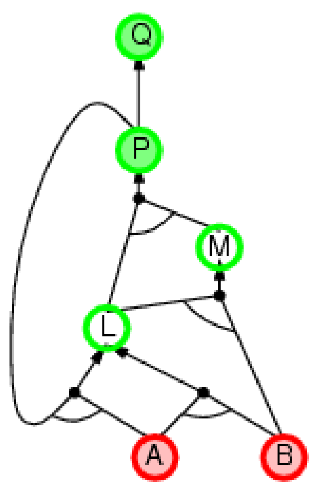
32

Backward chaining example



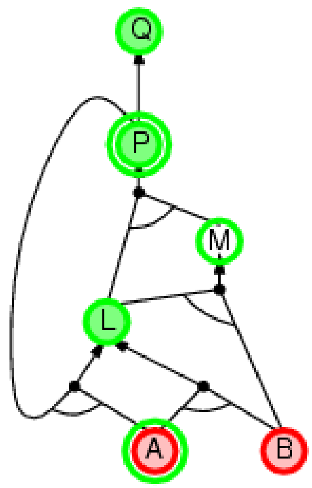
33

Backward chaining example



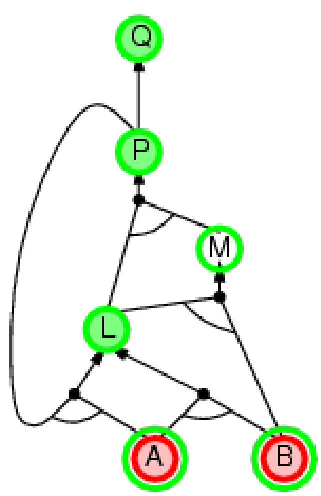
34

Backward chaining example



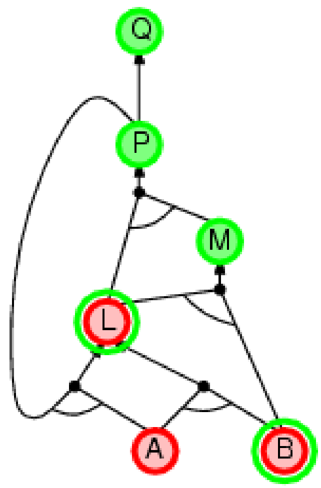
35

Backward chaining example



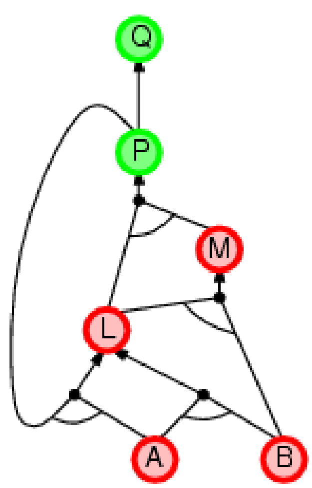
36

Backward chaining example



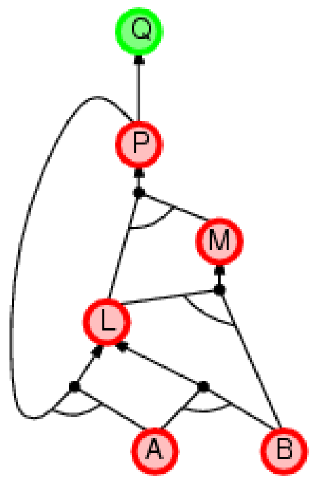
37

Backward chaining example



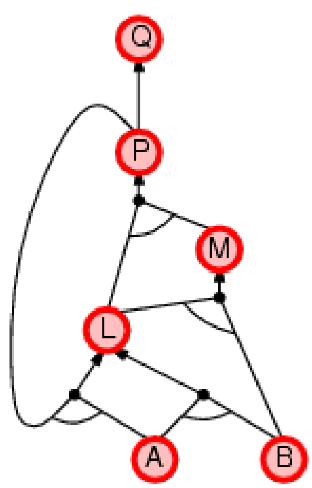
38

Backward chaining example



39

Backward chaining example



40

Forward vs. backward chaining

- FC is data-driven, automatic, unconscious processing,
  e.g., object recognition, routine decisions
- FC may do lots of work that is irrelevant to the goal
- BC is goal-driven, appropriate for problem-solving,
  e.g., How do I get an A in this class?
  e.g., What is my best exit strategy out of the classroom?
  e.g., How can I impress my date tonight?
- Complexity of BC can be much less than linear in size of KB

41

The DPLL algorithm

Determine if an input propositional logic sentence (in CNF) is satisfiable.

Improvements over truth table enumeration:

1. Early termination
   A clause is true if any literal is true.
   A sentence is false if any clause is false.

2. Pure symbol heuristic
   Pure symbol: always appears with the same "sign" in all clauses.
   e.g., In the three clauses $(A \vee \neg B)$, $(\neg B \vee \neg C)$, $(C \vee A)$, A and B are pure, C is impure.
   Make a pure symbol literal true.

3. Unit clause heuristic
   Unit clause: only one literal in the clause
   The only literal in a unit clause must be true.

42

The DPLL algorithm

function DPLL-SATISFIABLE?(s) **returns** *true* or *false*
  inputs: s, a sentence in propositional logic

  clauses ← the set of clauses in the CNF representation of s
  symbols ← a list of the proposition symbols in s
  return DPLL(*clauses*, *symbols*, [ ])

---

function DPLL(*clauses*, *symbols*, *model*) **returns** *true* or *false*

  if every clause in *clauses* is true in *model* **then return** *true*
  if some clause in *clauses* is false in *model* **then return** *false*
  P, *value* ← FIND-PURE-SYMBOL(*symbols*, *clauses*, *model*)
  if P is non-null **then return** DPLL(*clauses*, *symbols*–P, [P = *value*|*model*])
  P, *value* ← FIND-UNIT-CLAUSE(*clauses*, *model*)
  if P is non-null **then return** DPLL(*clauses*, *symbols*–P, [P = *value*|*model*])
  P ← FIRST(*symbols*); *rest* ← REST(*symbols*)
  return DPLL(*clauses*, *rest*, [P = *true*|*model*]) **or**
      DPLL(*clauses*, *rest*, [P = *false*|*model*])

43

Next Time

- WalkSAT
- Logical Agents: Wumpus
- First-Order Logic
- To Do:
  - Project #2
  - Finish Chapter 7
  - Start Chapter 8

44