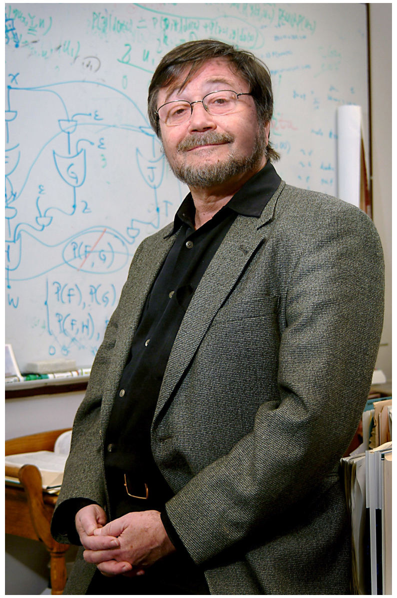
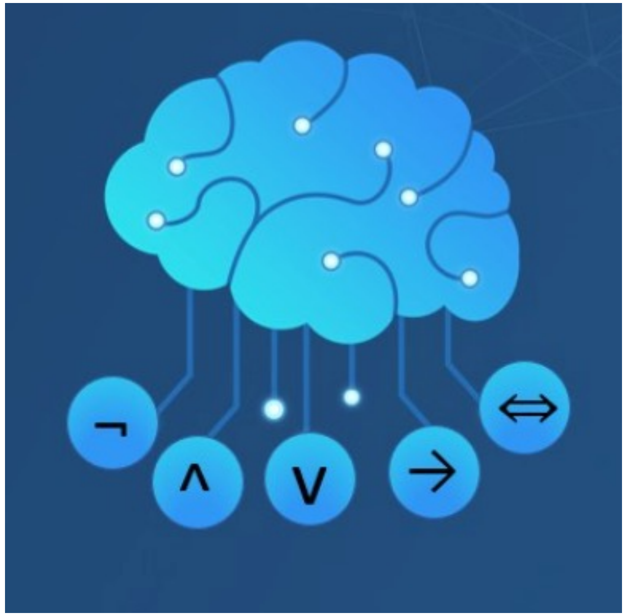


Artificial Intelligence



Propositional Logic (part 2)

CS 444 – Spring 2021

Dr. Kevin Molloy

Department of Computer Science

James Madison University

JMU JAMES MADISON UNIVERSITY

Much of this lecture is taken from
Dan Klein and Pieter Abbeel AI class at UC Berkeley

Proof Methods

Proof methods divide into (roughly) two kinds:

Model checking:

- Truth table enumeration (always exponential in n)
- Improved backtracking, e.g., Davis-Putnam-Logemann-Loveland
- Backtracking with constraint propagation, backjumping
- Heuristic search in model space (sound but incomplete) e.g., min-conflicts, etc.

Theorem Proving/Deductive Systems: Application of inference rules

- Legitimate (sound) generation of new sentences from old
- Proof = a sequence of inference rule applications
- Typically requires translation of sentence into a **normal form**

Logical Equivalence

Two sentences are logically equivalent iff true in the same models.

$\alpha \equiv \beta$ iff $a \vDash \beta$ and $\beta \vDash \alpha$

$(\alpha \wedge \beta) \equiv (\beta \wedge \alpha)$	Commutativity of $\wedge$
$(\alpha \vee \beta) \equiv (\beta \vee \alpha)$	Commutativity of $\vee$
$((\alpha \wedge \beta) \wedge \gamma) \equiv (\alpha \wedge (\beta \wedge \gamma))$	Associativity of $\wedge$
$((\alpha \vee \beta) \vee \gamma) \equiv (\alpha \vee (\beta \vee \gamma))$	Associativity of $\vee$
$\neg(\neg\alpha) \equiv \alpha$	Double negation elimination
$(\alpha \Longrightarrow \beta) \equiv (\neg\beta \Longrightarrow \neg\alpha)$	Contrapositive
$(\alpha \Longrightarrow \beta) \equiv (\neg\alpha \vee \beta)$	Implication elimination
$(\alpha \Leftrightarrow \beta) \equiv ((\alpha \Longrightarrow \beta) \wedge (\beta \Longrightarrow \alpha))$	Biconditional elimination
$\neg(\alpha \wedge \beta) \equiv (\neg\alpha \vee \neg \beta)$	de Morgan
$\neg(\alpha \vee \beta) \equiv (\neg\alpha \wedge \neg\beta)$	de Morgan
$(\alpha \wedge (\beta \vee \gamma)) \equiv ((\alpha \wedge \beta) \vee (\alpha \wedge \gamma))$	Distribution $\wedge$ of over $\vee$
$(\alpha \vee (\beta \wedge \gamma)) \equiv ((\alpha \vee \beta) \wedge (\alpha \vee \gamma))$	Distribution $\vee$ of over $\wedge$

Validity and Satisfiability

A sentence is **valid** if it is true in **all** models.

e.g., True, $A \vee \neg A$, $A \Longrightarrow A$, $(A \wedge (A \Longrightarrow B)) \Longrightarrow B$

Validity is connected to inference via the Deduction Theorem:

$KB \vDash \alpha$ iff $(KB \Longrightarrow \alpha)$ 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 model. e.g., $A \wedge \neg A$

Satisfiability is connected to inference via the following:

$KB \vDash \alpha$ iff $(KB \wedge \neg\alpha)$ is unsatisfiable

i.e., prove α by reduction ad absurdum (by contradiction)

Deductive Systems: Rules of Inference

Modus ponens or **implication-elimination** (form an implication and the premise of the implication, you can infer the solution):

$$\frac{\alpha \Rightarrow \beta, \alpha}{\beta}$$

And-elimination (from a conjunction, you can infer any of the conjuncts):

$$\frac{\alpha_1 \land \alpha_2 \land \ldots \land \alpha_n}{\alpha_i}$$

And-introduction (from a list of sentences, you can infer their conjunction)

$$\frac{\alpha_1, \alpha_2, \ldots, \alpha_n}{\alpha_1 \land \alpha_2 \land \ldots \land \alpha_n}$$

Double negation elimination:

$$\frac{\neg\neg\alpha}{\alpha}$$

Unit resolution (from a disjunction, if one of the disjuncts if false, you can infer the other is true)

$$\frac{\alpha \lor \beta, \neg\beta}{\alpha}$$

Resolution: Since β can not be true and false, one of the other disjuncts must be true in one of the premises.

$$\frac{\alpha \lor \beta, \neg\beta \lor \gamma}{\alpha \lor \gamma} \qquad \frac{\neg\alpha \Rightarrow \beta,\ \beta \Rightarrow \gamma}{\neg\alpha \Rightarrow \gamma}$$

Inference by Resolution

Conjunction Normal Form (CNF – universal)

Conjunction of **disjunctions of literals**

e.g. (A ∨ ¬B) ∧ (B ∨ ¬C ∨ ¬D)

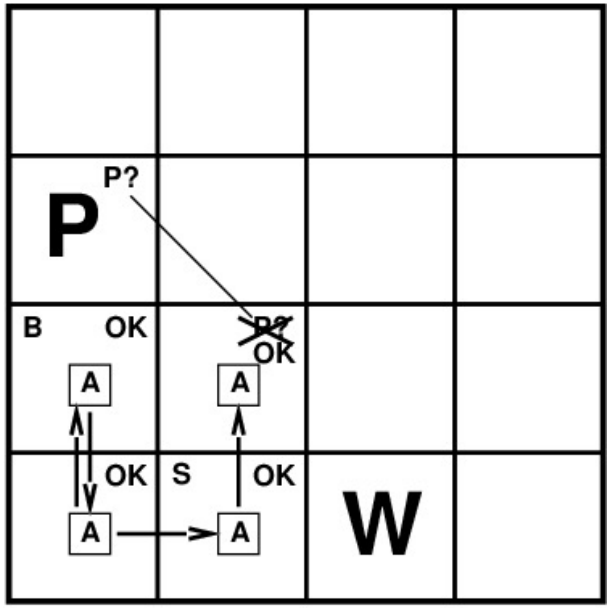
Resolution inference rule (for CNF): complete for propositional logic

$$\frac{\ell_1 \vee \cdots \vee \ell_k, \quad m_1 \vee \ldots \vee m_n}{\ell_1 \vee \cdots \vee \ell_{i-1} \vee \ell_{i+1} \vee \ldots \ell_k \vee m_1 \vee \ldots \vee m_{j-1} \vee m_{j+1} \vee \cdots \vee m_n}$$

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



Conversion to CNF

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

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

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

2. Eliminate $\Longrightarrow$, replacing $\alpha \Longrightarrow \beta$ 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 and flatten

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

Resolution Algorithm

```
function PL-Resolution(KB, α) returns true/false
    Input:    KB, the knowledge base, a sentence in propositional logic
              α, the query, a sentence in propositional logic
    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  U resolvents

          if new ⊆ clauses then return false
          clauses ← clauses U new
```

Resolution Example

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

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

Want to prove KB ^ ¬ α is a contradiction. $((B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})) \wedge \neg B_{1,1} \wedge P_{1,2}$

Step 1) Convert this clause to CNF:

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

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

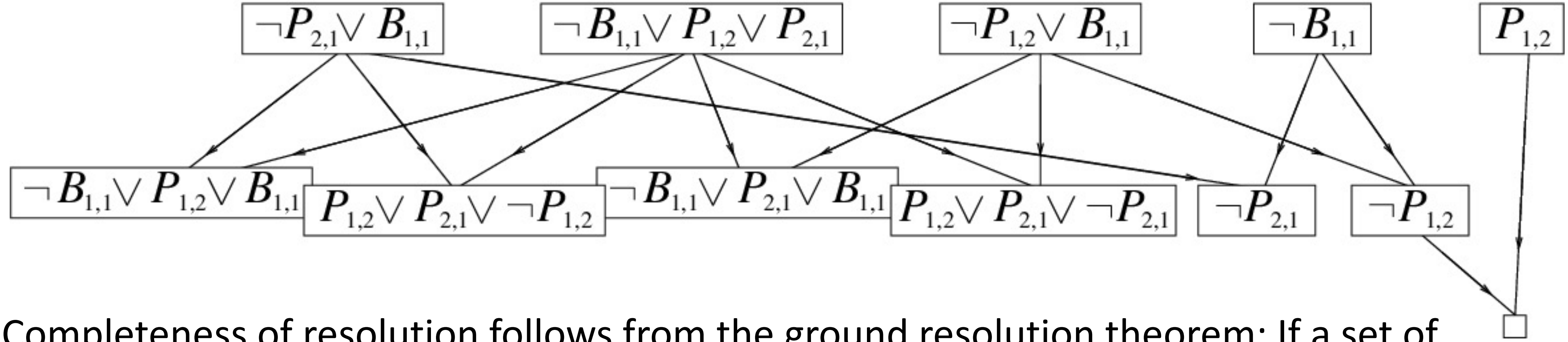
$$(\neg B_{1,1} \vee (P_{1,2} \vee P_{2,1})) \wedge (\neg(P_{1,2} \vee P_{2,1}) \vee B_{1,1}) \wedge \neg B_{1,1} \wedge \neg P_{1,2}$$

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

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

Resolution Example

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



Completeness of resolution follows from the ground resolution theorem: If a set of clauses S is unsatisfiable, then the resolution closure RC(S) of those clauses contains an empty clause.

RC(S): set of all clauses derivable by repeated application of resolution rule to clauses in S or their derivatives.

Definite Clauses and Horn Clauses

Inference by resolution is complete, but sometimes an overkill

Definite clause: disjunction of literals of which exactly one is positive.

$\neg L_{1,1} \vee B_{1,1}$ is a definite clause $P_{1,2} \vee P_{1,2}$ is not a definite clause

Horn clause: disjunction of literals of which at most one is positive.

$\neg L_{1,1} \vee B_{1,1}$ is a horn clause $P_{1,2} \vee P_{1,2}$ is not a horn clause

Negate literals $\neg A$ rewritten as ($A \Longrightarrow$ False) (integrity constraints)

Inference with Horn clauses can be done through forward chaining and backward chaining

These are more efficient than the resolution algorithm, **runs in linear time**

Horn Form and Forwards/Backwards Chaining

Horn Form (restricted) KB (= conjunction of Horn clauses)

e.g., C ∧ (B ⟹ A) ∧ (C ∧ D⟹ B)

Modus Ponens: complete form Horn KBs

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

Known as forward chaining inference rule; repeated applications until sentence of interest obtained – forward chaining algorithm.

Modus Tollens: a form of Modus Ponens

$$\frac{\neg\beta, \quad \alpha_1 \wedge \ \ldots \wedge \ \alpha_n \Rightarrow \ \beta}{\neg(\alpha_1 \wedge \ \ldots \wedge \ \alpha_n)}$$

Known as backward chaining inference rule, repeated applications until all premises obtained – backward chaining algorithm.

Both algorithms run in **linear** time

Forward Chaining

Idea: add literals in KB to facts (satisfied premises)

apply each premise satisfied in KB (fire rules)

add rule's conclusion as new fact/premise to the KB

(this is inference propagation via forward checking).

stop when query found as fact or no more inferences.

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

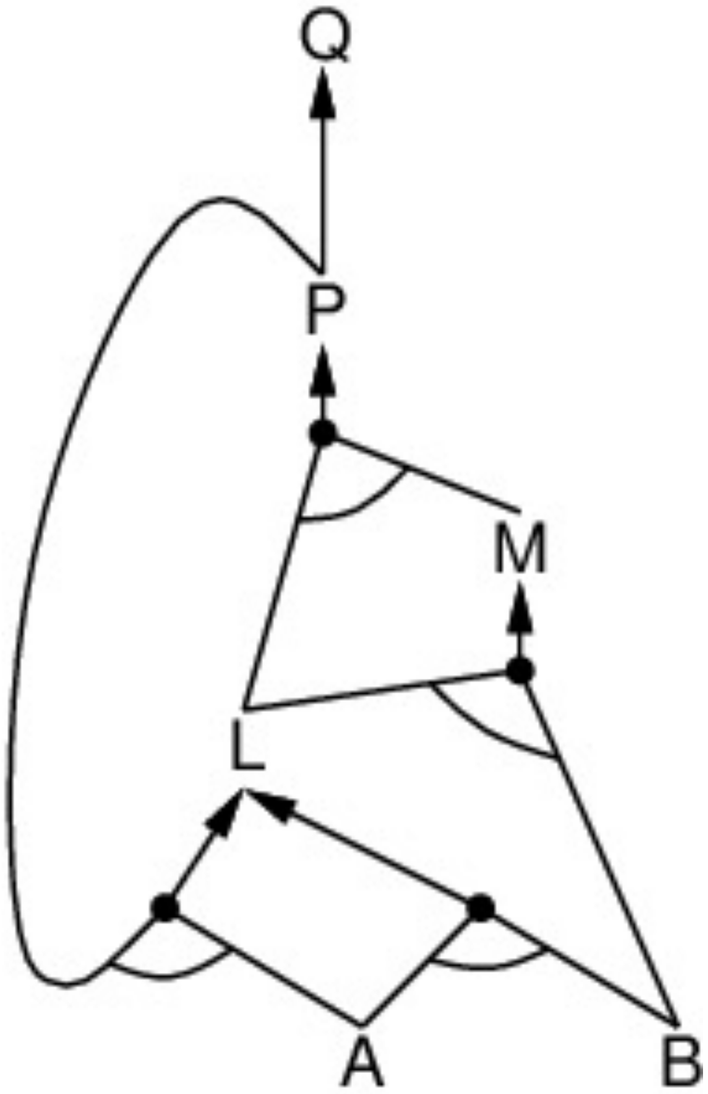
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

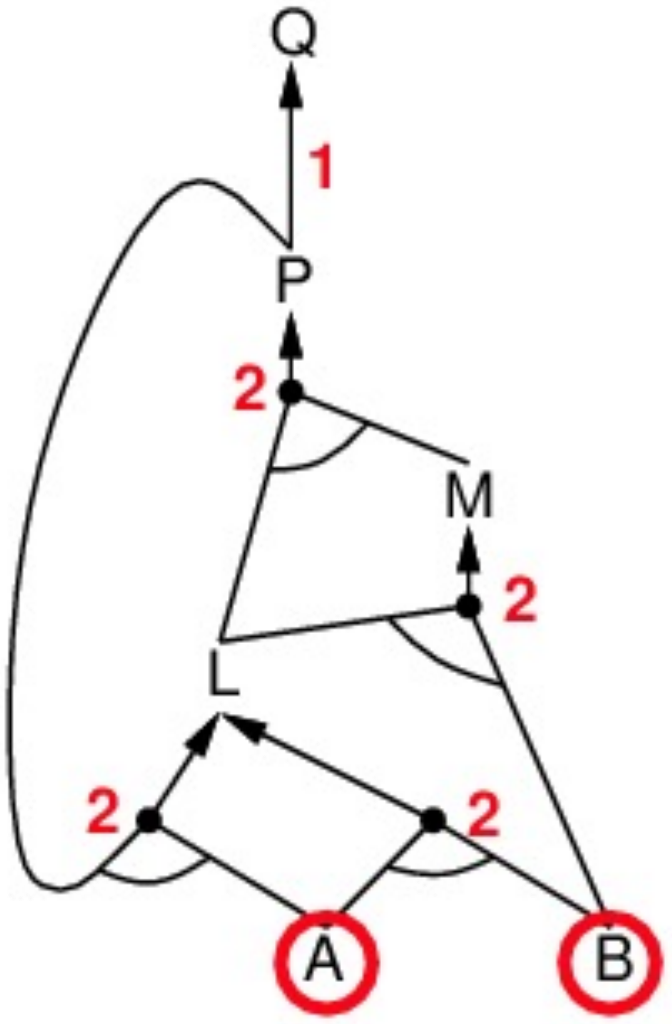
$A \wedge B \Rightarrow L$

A

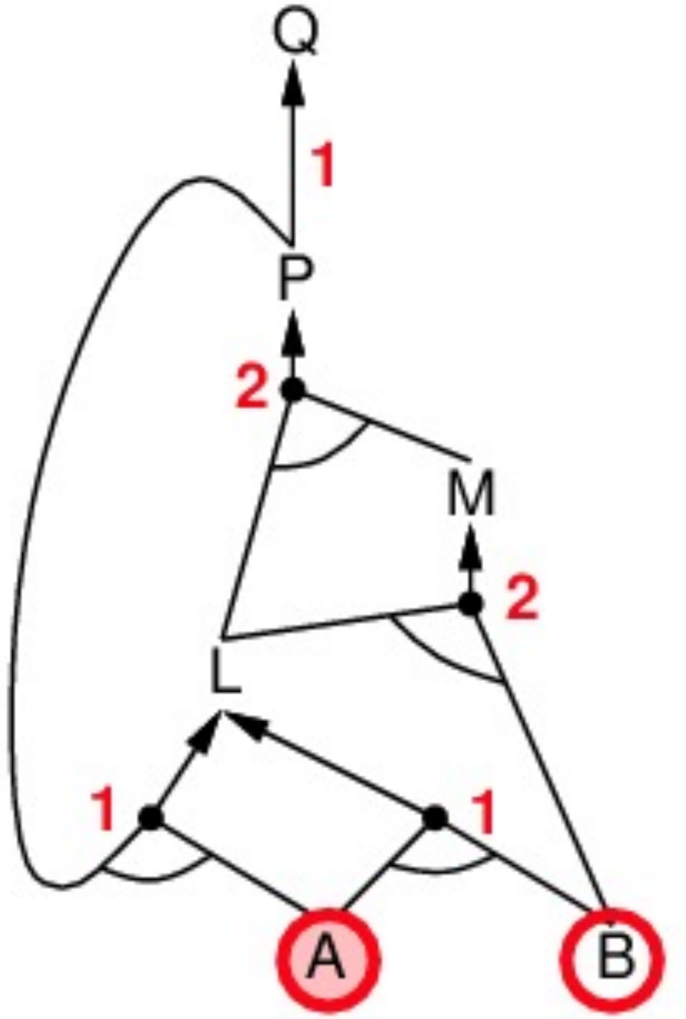
B



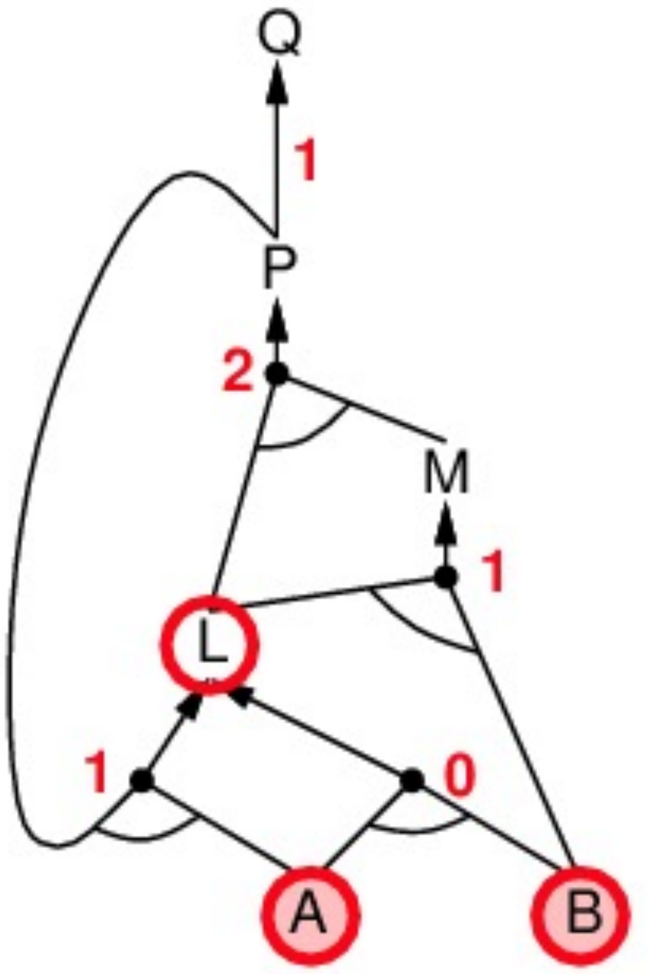
Forward Chaining Example



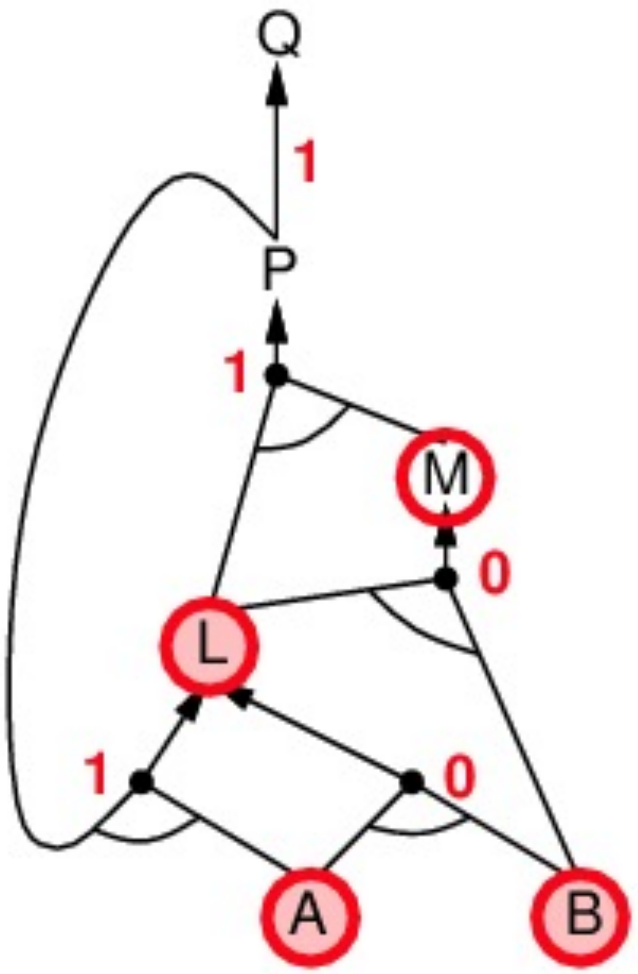
Forward Chaining Example



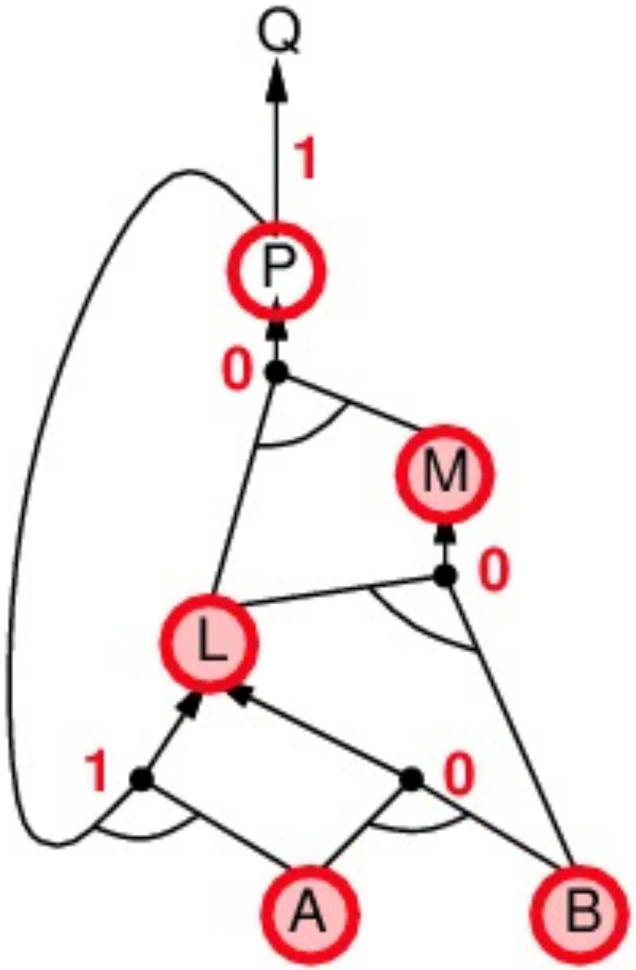
Forward Chaining Example



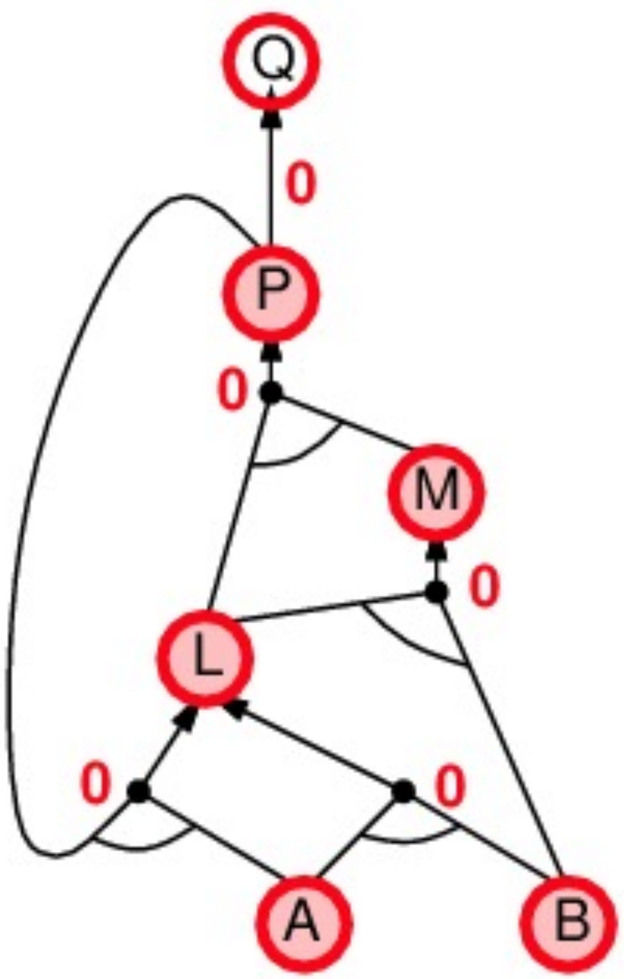
Forward Chaining Example



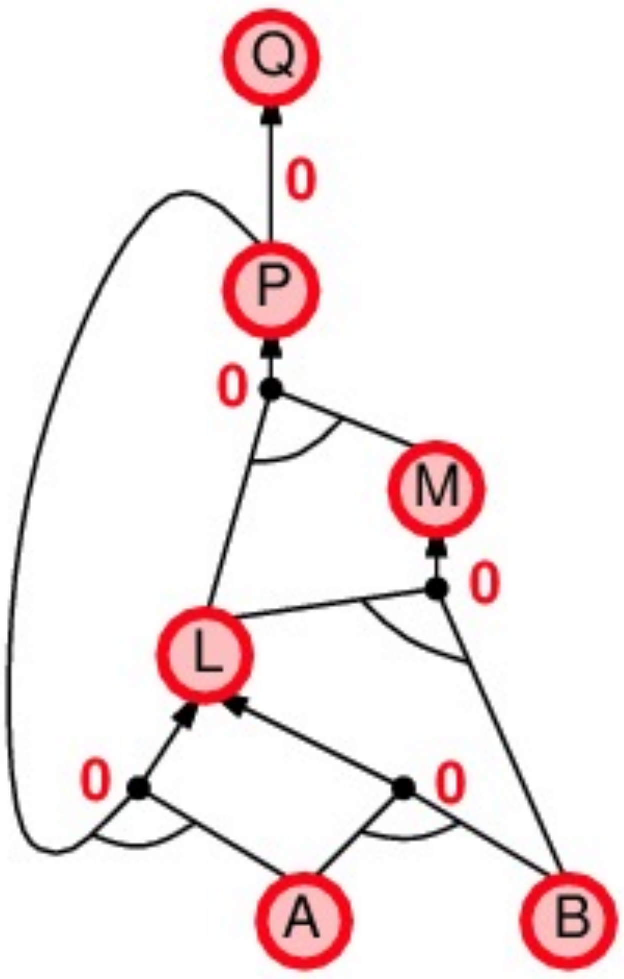
Forward Chaining Example



Forward Chaining Example



Forward Chaining Example



Forward Chaining – 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***.

Proof: Suppose a clause $a_1 \wedge \ldots \wedge a_k \Longrightarrow b$ is false in m.

We know that $a_1 \wedge \ldots \wedge a_k$ must be true, so b must be false. But that contradicts that we have reached a fixed point. Hence:

4) *m* is a model of KB
5) KB ⊨ *q*, *q* is true in every mode of KB, including ***m***

Backward Chaining

Idea: goal-driven reasoning – 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

Comparing FC and BC

FC is data-driven, 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, because only relevant facts are touched.

Inference-based Agent in Wumpus World

A wumpus-world agent using propositional logic:

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

$$\neg W_{1,1}$$

$$B_{x,y} \Leftrightarrow (P_{x,y+1} \vee P_{x,y-1} \vee P_{x+1,y} \vee P_{x-1,y})$$

$$S_{x,y} \Leftrightarrow (W_{x,y+1} \vee W_{x,y-1} \vee W_{x+1,y} \vee W_{x-1,y})$$

$$W_{1,1} \vee W_{1,2} \vee \ldots \vee W_{4,4}$$

$$\neg W_{1,1} \vee \neg W_{1,2}$$

$$\neg W_{1,1} \vee \neg W_{1,3}$$

...

64 distinct proposition symbols, 155 sentences.

Propositional Logic Summary

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 wrt models
- **Entailment**: necessary truth of one sentence given another
- **Inference**: deriving sentences from other sentences
- **Soundness**: derivations produce only entailed sentences
- **Completeness**: derivations can produce all entailed sentences

Wumpus world requires the ability to represent partial and negated information, reason by cases, etc.

Propositional logic does not scale to environments of unbounded size, as it lacks expressive power to deal concisely with time, space, and universal patterns of relationships among objects.