

Chapter 7: Logical Agents

Propositional Logic

Knowledge-based agents

a conclusion reached on the basis of evidence and reasoning.



- Knowledge base (KB) = set of sentences in a formal language
- Declarative approach to building an agent (or other system):
 - Tell it what it needs to know
- Then it can ask itself what to do - answers should follow from the KB
- Distinction between data and program
- Fullest realization of this philosophy was in the field of expert systems or knowledge-based systems in the 1970s and 1980s

What is logic?

- **Logic** is a formal system for manipulating facts so that true conclusions may be drawn
 - “The tool for distinguishing between the true and the false” – Averroes (12th cen.)
- **Syntax:** rules for constructing valid sentences
 - E.g., $x + 2 \geq y$ is a valid arithmetic sentence, $\geq x 2 y +$ is not
- **Semantics:** “meaning” of sentences, or relationship between logical sentences and the real world
 - Specifically, semantics defines truth of sentences
 - E.g., $x + 2 \geq y$ is true in a world where $x = 5$ and $y = 7$

Overview

- Propositional logic
- Inference rules and theorem proving
- First order logic

Propositional logic: Syntax

- **Atomic sentence:**
 - A *proposition symbol* representing a true or false statement
- **Negation:**
 - If P is a sentence, $\neg P$ is a sentence
- **Conjunction:**
 - If P and Q are sentences, $P \wedge Q$ is a sentence
- **Disjunction:**
 - If P and Q are sentences, $P \vee Q$ is a sentence
- **Implication:**
 - If P and Q are sentences, $P \Rightarrow Q$ is a sentence
- **Biconditional:**
 - If P and Q are sentences, $P \Leftrightarrow Q$ is a sentence
- $\neg, \wedge, \vee, \Rightarrow, \Leftrightarrow$ are called *logical connectives*

Propositional logic: Semantics

- A **model** specifies the true/false status of each proposition symbol in the knowledge base
 - E.g., **P** is true, **Q** is true, **R** is false
 - With three symbols, there are 8 possible models, and they can be enumerated exhaustively
- Rules for evaluating truth with respect to a model:

	$\neg P$ is true	iff	P is false	
AND	$P \wedge Q$ is true	iff	P is true and Q is true	
OR	$P \vee Q$ is true	iff	P is true or Q is true	
	$P \Rightarrow Q$ is true	iff	P is false or Q is true	
	$P \Leftrightarrow Q$ is true	iff	$P \Rightarrow Q$ is true and $Q \Rightarrow P$ is true	

Truth tables

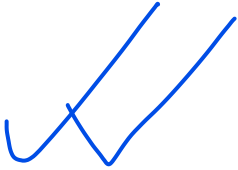
- A **truth table** specifies the truth value of a composite sentence for each possible assignments of truth values to its atoms

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$	$P \Leftrightarrow Q$
<i>false</i>	<i>false</i>	<i>true</i>	<i>false</i>	<i>false</i>	<i>true</i>	<i>true</i>
<i>false</i>	<i>true</i>	<i>true</i>	<i>false</i>	<i>true</i>	<i>true</i>	<i>false</i>
<i>true</i>	<i>false</i>	<i>false</i>	<i>false</i>	<i>true</i>	<i>false</i>	<i>false</i>
<i>true</i>	<i>true</i>	<i>false</i>	<i>true</i>	<i>true</i>	<i>true</i>	<i>true</i>

- The **truth value** of a more **complex sentence** can be evaluated *recursively* or *compositionally*

Logical equivalence

- Two sentences are **logically equivalent** iff they are true in same models


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

Validity, satisfiability

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

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

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., $A \wedge \neg A$

Entailment

act or fact of entailing, or involving by necessity or as a consequence

- **Entailment** means that a sentence follows from the premises contained in the knowledge base:

$$KB \models \alpha$$

- Knowledge base KB entails sentence α if and only if α is true in all models where KB is true

- E.g., $x = 0$ entails $x * y = 0$
- Can α be true when KB is false?

- $KB \models \alpha$ iff $(KB \Rightarrow \alpha)$ is *valid*
- $KB \models \alpha$ iff $(KB \wedge \neg \alpha)$ is *unsatisfiable*



Inference

- **Logical inference:** a procedure for generating sentences that follow from a knowledge base KB
- ✓ • An inference procedure is **sound** if whenever it derives a sentence α , $KB \models \alpha$
 - A sound inference procedure can derive only true sentences
- An inference procedure is **complete** if whenever $KB \models \alpha$, α can be derived by the procedure
 - ✓ – A complete inference procedure can derive every entailed sentence

Inference

- How can we **check** whether a sentence α is entailed by KB?
- How about we enumerate all possible models of the KB (truth assignments of all its symbols), and check that α is true in every model in which KB is true?
 - Is this sound?
 - Is this complete?
- ✓ • Problem: if KB contains **n symbols**, the truth table will be of size **2^n**
- ✓ • **Better idea**: use *inference rules*, or sound procedures to generate new sentences or *conclusions* given the *premises* in the KB

Inference rules

- Modus Ponens

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

← premises

← conclusion

- And-elimination

$$\frac{\alpha \wedge \beta}{\alpha}$$

Inference rules

- And-introduction

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

- Or-introduction

$$\frac{\alpha}{\alpha \vee \beta}$$

Inference rules

- Double negative elimination

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

- Unit resolution

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

Resolution

$$\frac{\alpha \vee \beta, \neg\beta \vee \gamma}{\alpha \vee \gamma}$$

or

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

- Example:

α : “The weather is dry”

β : “The weather is rainy”

γ : “I carry an umbrella”

Resolution is complete

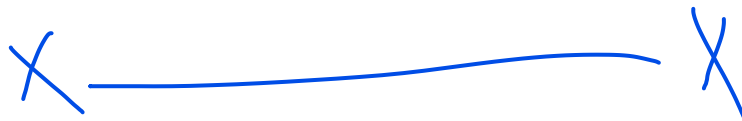
$$\frac{\alpha \vee \beta, \neg \beta \vee \gamma}{\alpha \vee \gamma}$$

$\models B \rightarrow \alpha$
 $\neg \models B \vee \alpha$

- To prove $KB \models \alpha$, assume $KB \wedge \neg \alpha$ and derive a contradiction \perp
 - Rewrite $KB \wedge \neg \alpha$ as a conjunction of *clauses*, or disjunctions of *literals*
 - *Conjunctive normal form* (CNF)
 - Keep applying resolution to clauses that contain *complementary literals* and adding resulting clauses to the list
 - If there are no new clauses to be added, then KB does not entail α
 - If two clauses resolve to form an *empty clause*, we have a contradiction and $KB \models \alpha$

Complexity of inference

- Propositional inference is **co-NP-complete**
 - Complement of the SAT problem: $\alpha \models \beta$ if and only if the sentence $\alpha \wedge \neg \beta$ is **unsatisfiable**
 - Every known inference algorithm has **worst-case exponential running time**
- Efficient inference possible for restricted cases



Definite clauses

and
 \wedge

- A **definite clause** is a **disjunction** with **exactly one positive literal**
- Equivalent to $(P_1 \wedge \dots \wedge P_n) \Rightarrow Q$

premise or body conclusion or head
- Basis of logic programming (**Prolog**)
- **Efficient (linear-time)** complete inference through *forward chaining* and *backward chaining*



Forward chaining

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

$$P \Rightarrow Q$$

$$L \wedge M \Rightarrow P$$

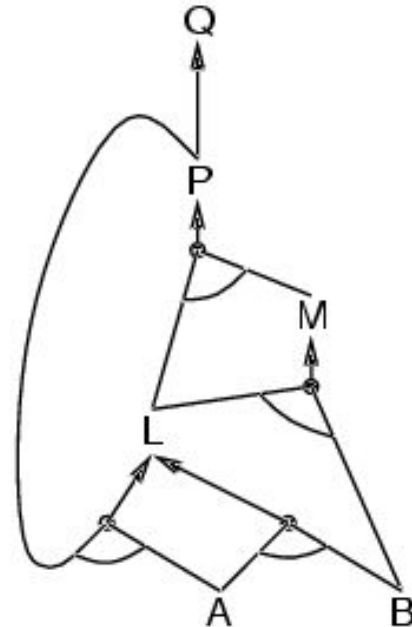
$$B \wedge L \Rightarrow M$$

$$A \wedge P \Rightarrow L$$

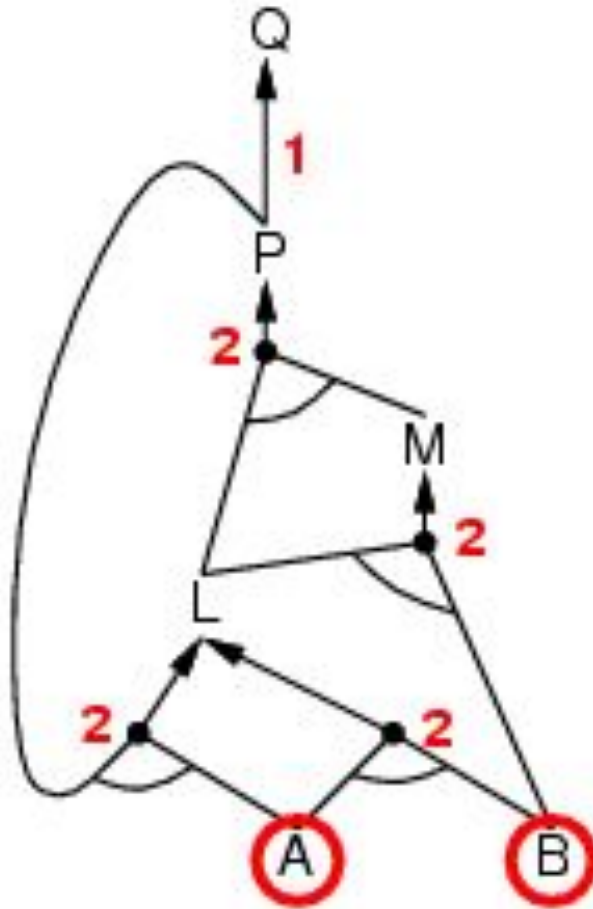
$$A \wedge B \Rightarrow L$$

A

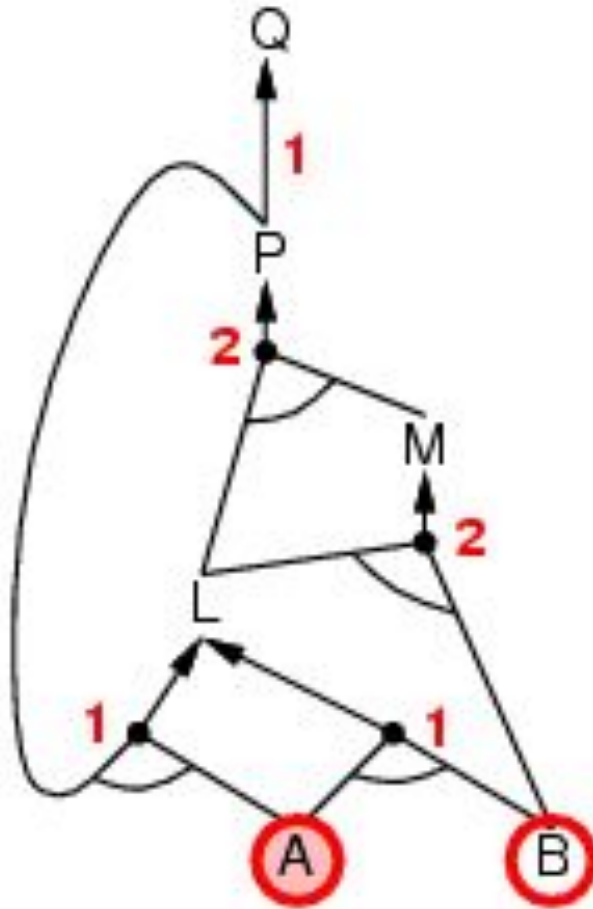
B



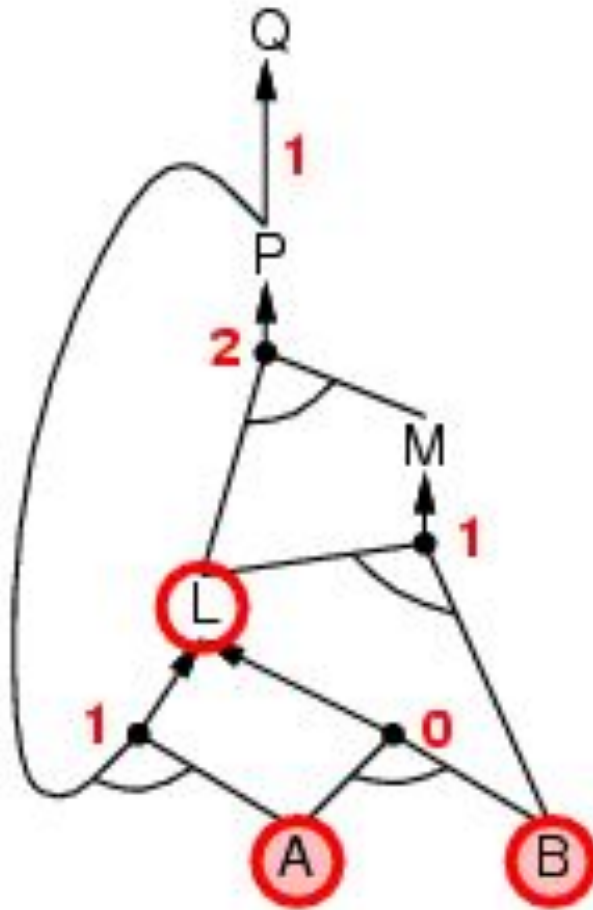
Forward chaining example



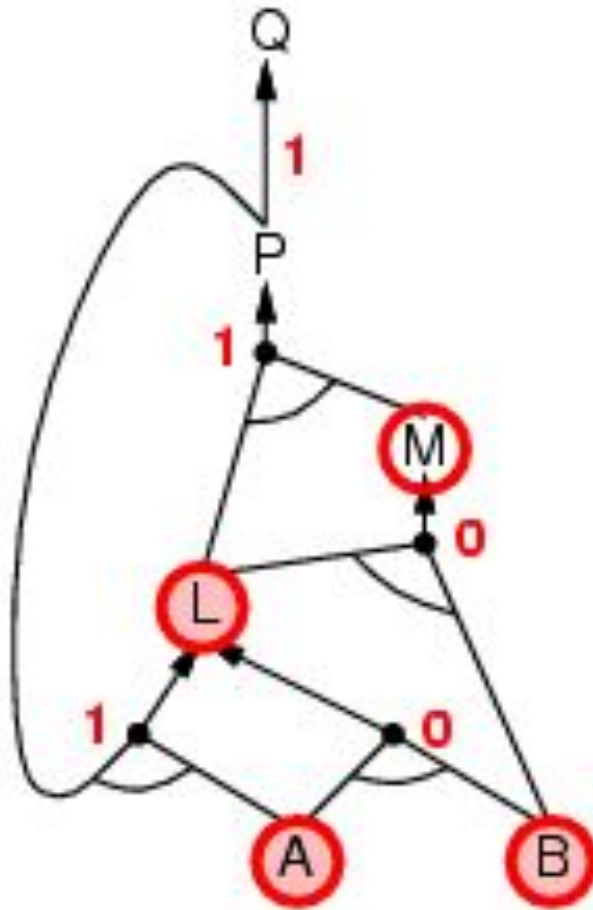
Forward chaining example



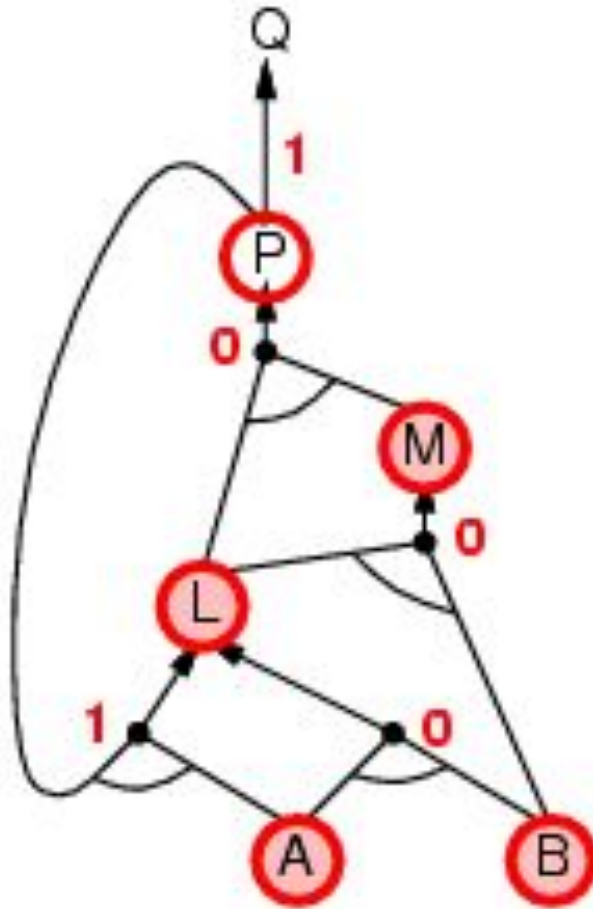
Forward chaining example



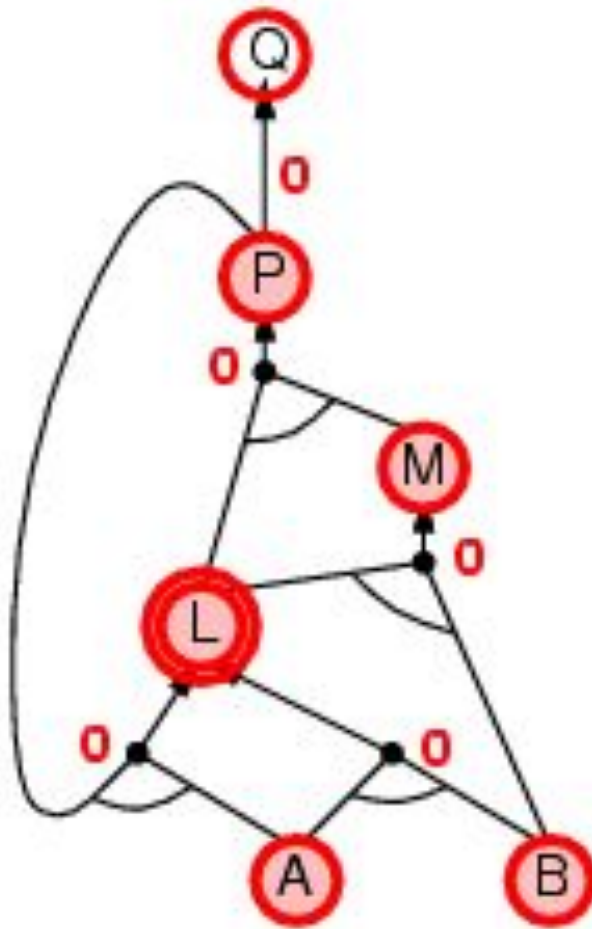
Forward chaining example



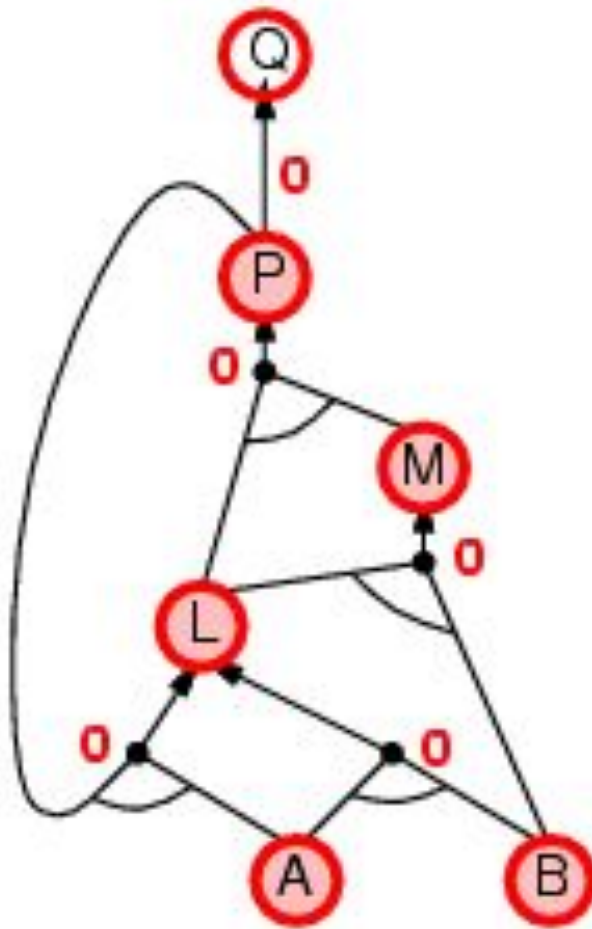
Forward chaining example



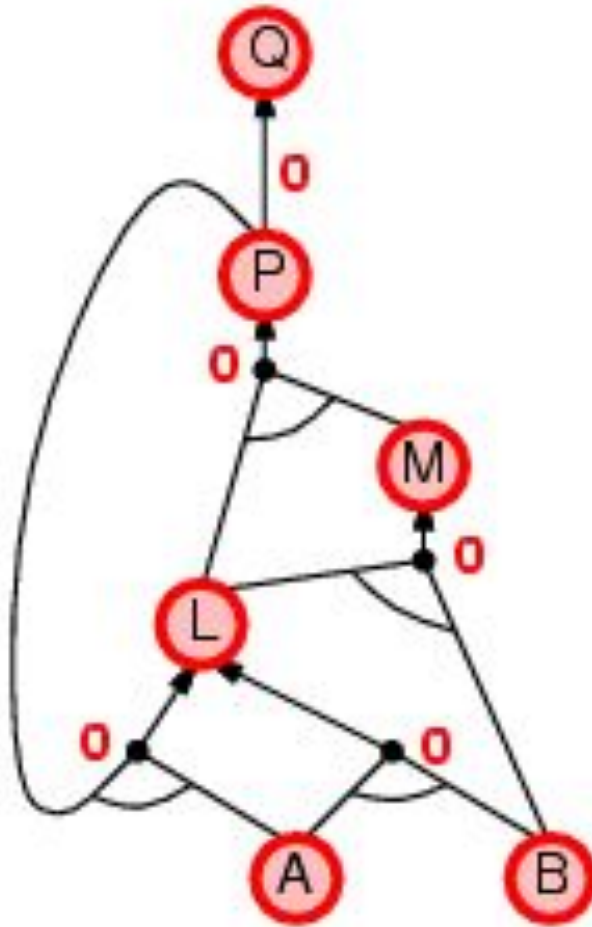
Forward chaining example

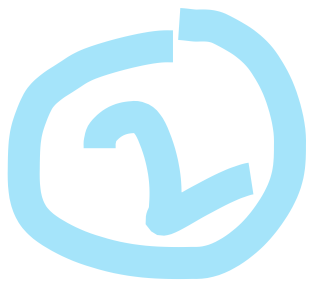


Forward chaining example



Forward chaining example

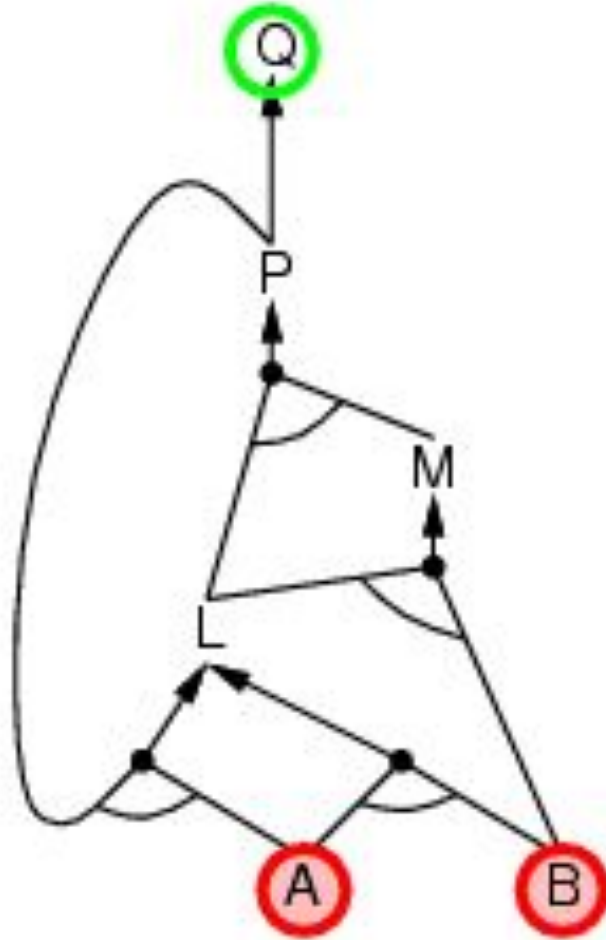




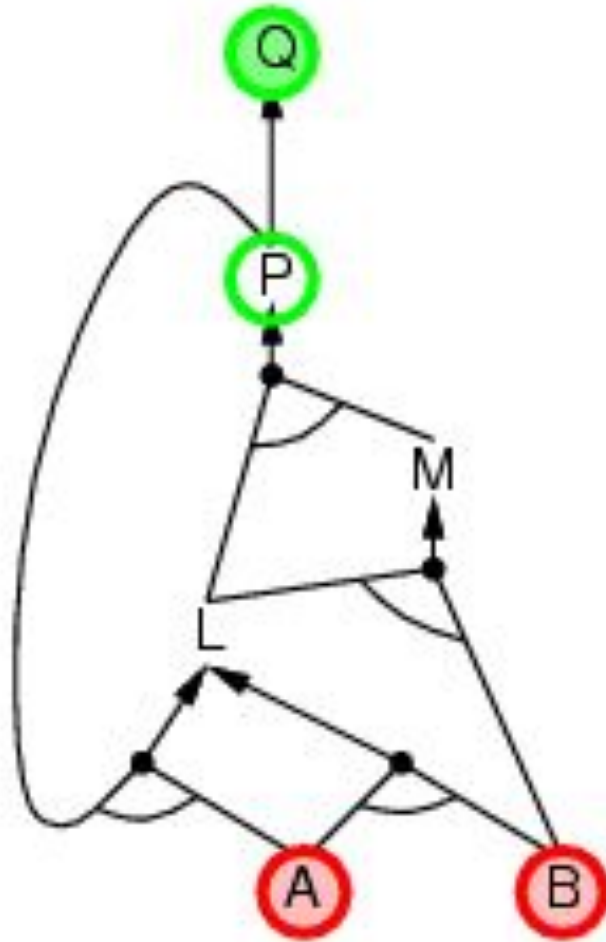
Backward chaining

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

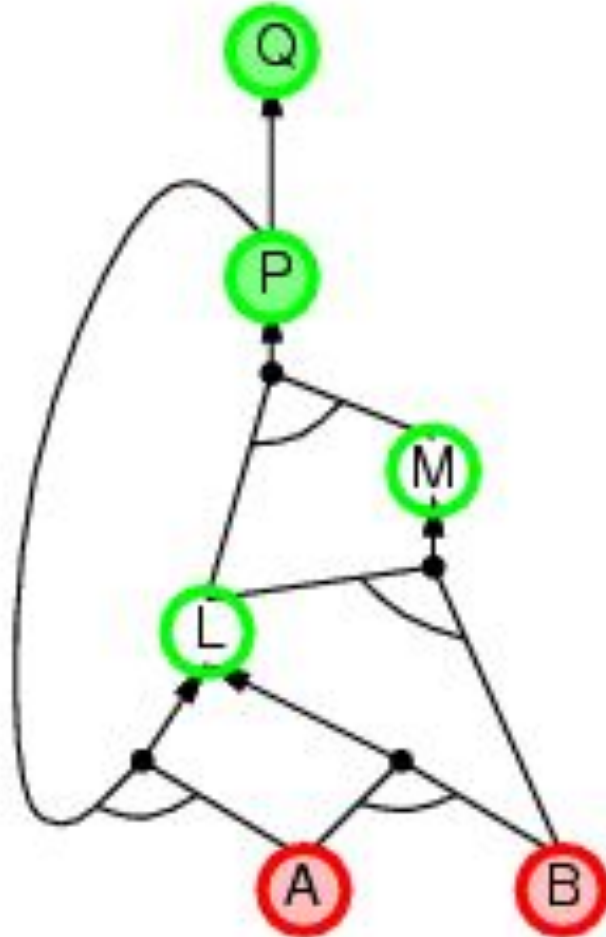
Backward chaining example



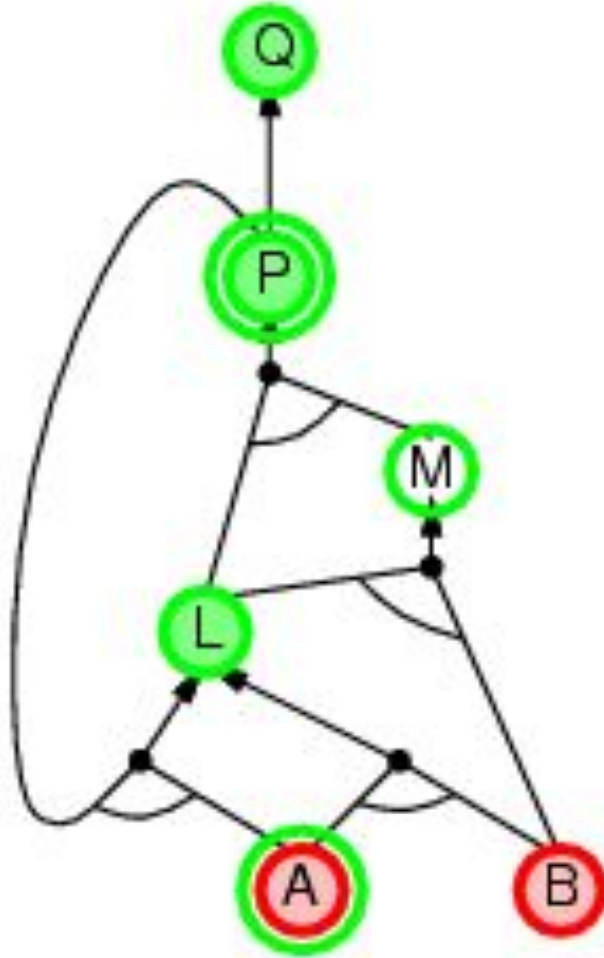
Backward chaining example



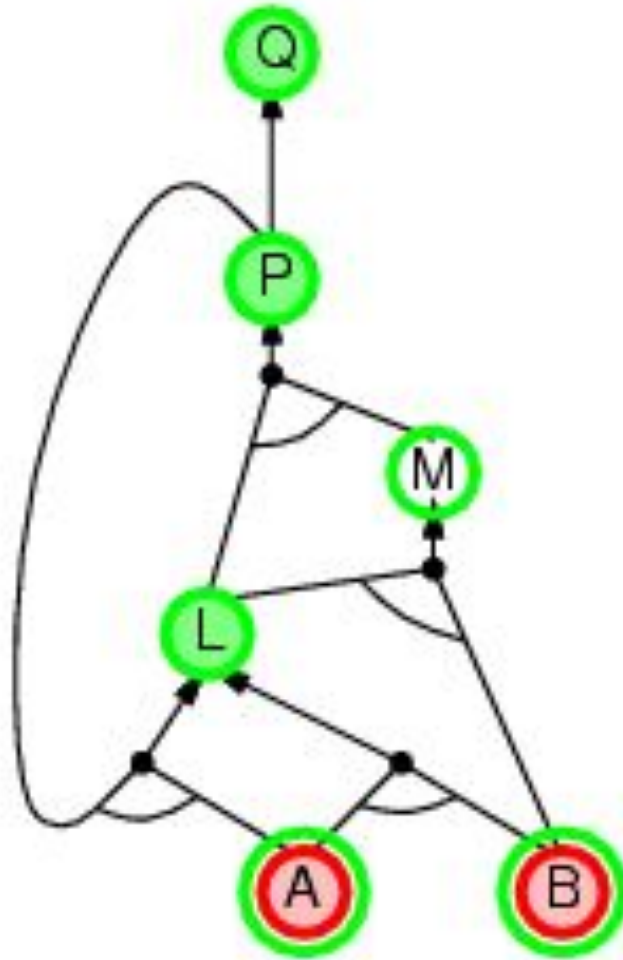
Backward chaining example



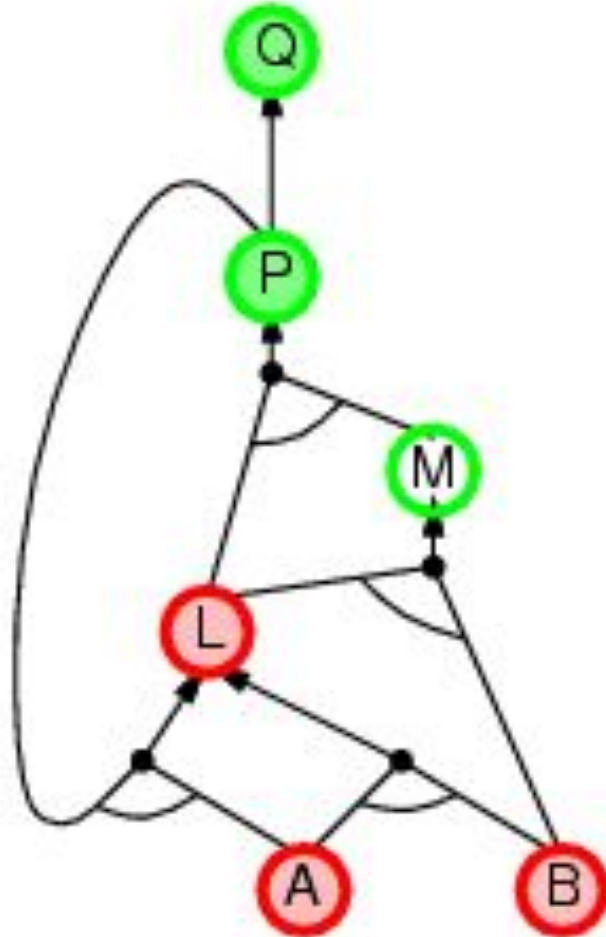
Backward chaining example



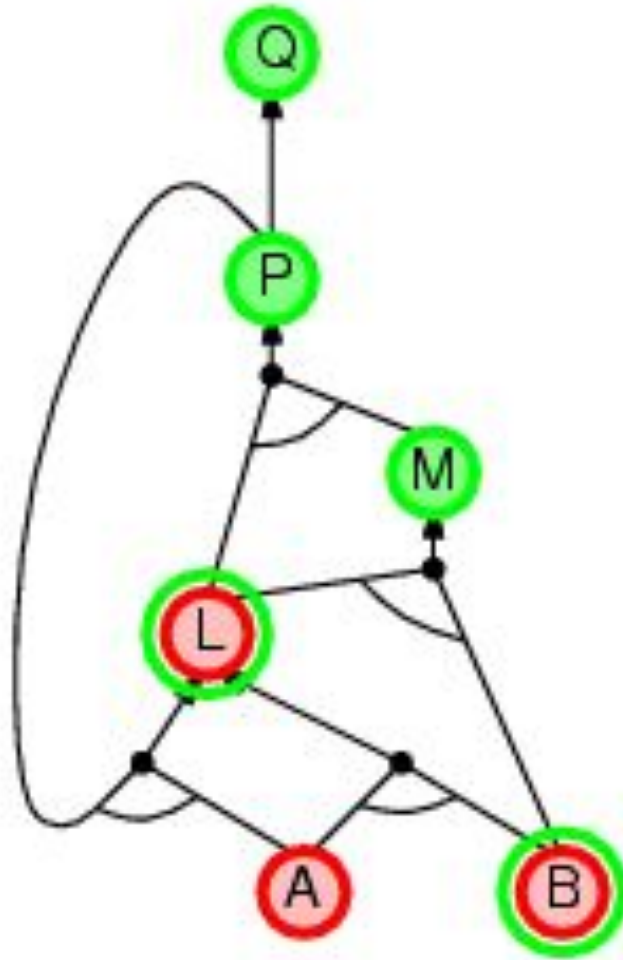
Backward chaining example



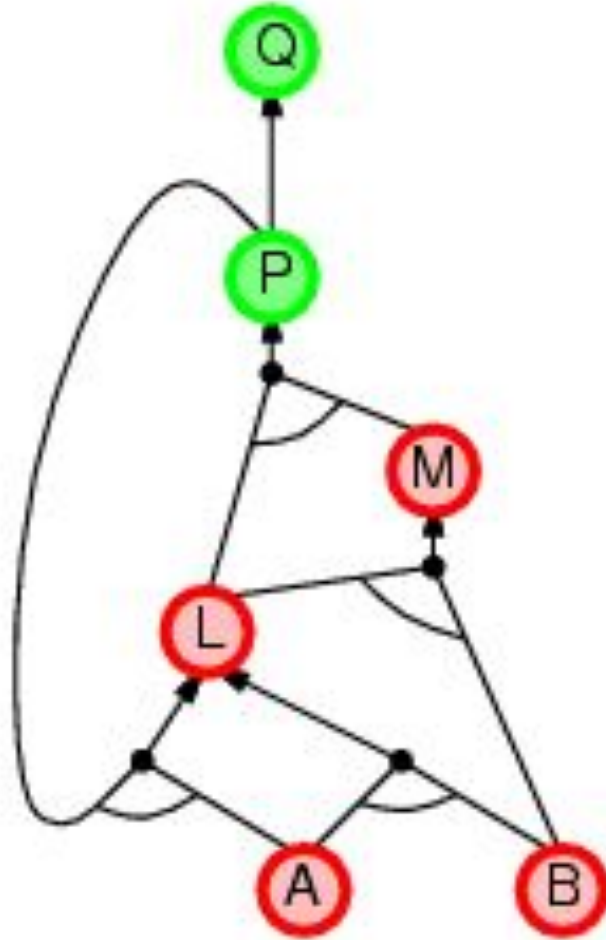
Backward chaining example



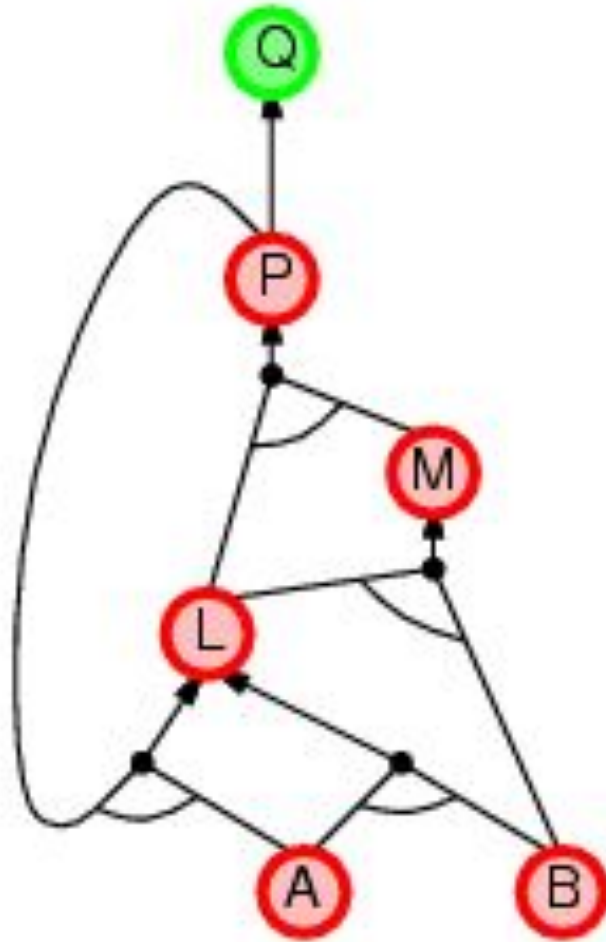
Backward chaining example



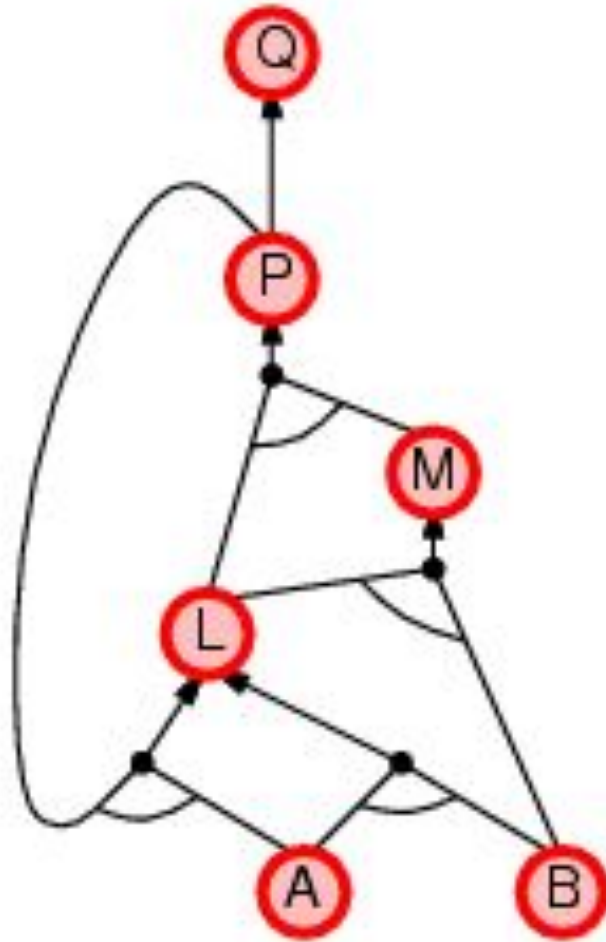
Backward chaining example



Backward chaining example



Backward chaining example



Forward vs. backward chaining

- Forward chaining is data-driven, automatic processing
 - May do lots of work that is irrelevant to the goal
- Backward chaining is goal-driven, appropriate for problem-solving
 - Complexity can be much less than linear in size of KB

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

- Resolution is complete for propositional logic
- Forward, backward chaining are linear-time, complete for definite clauses