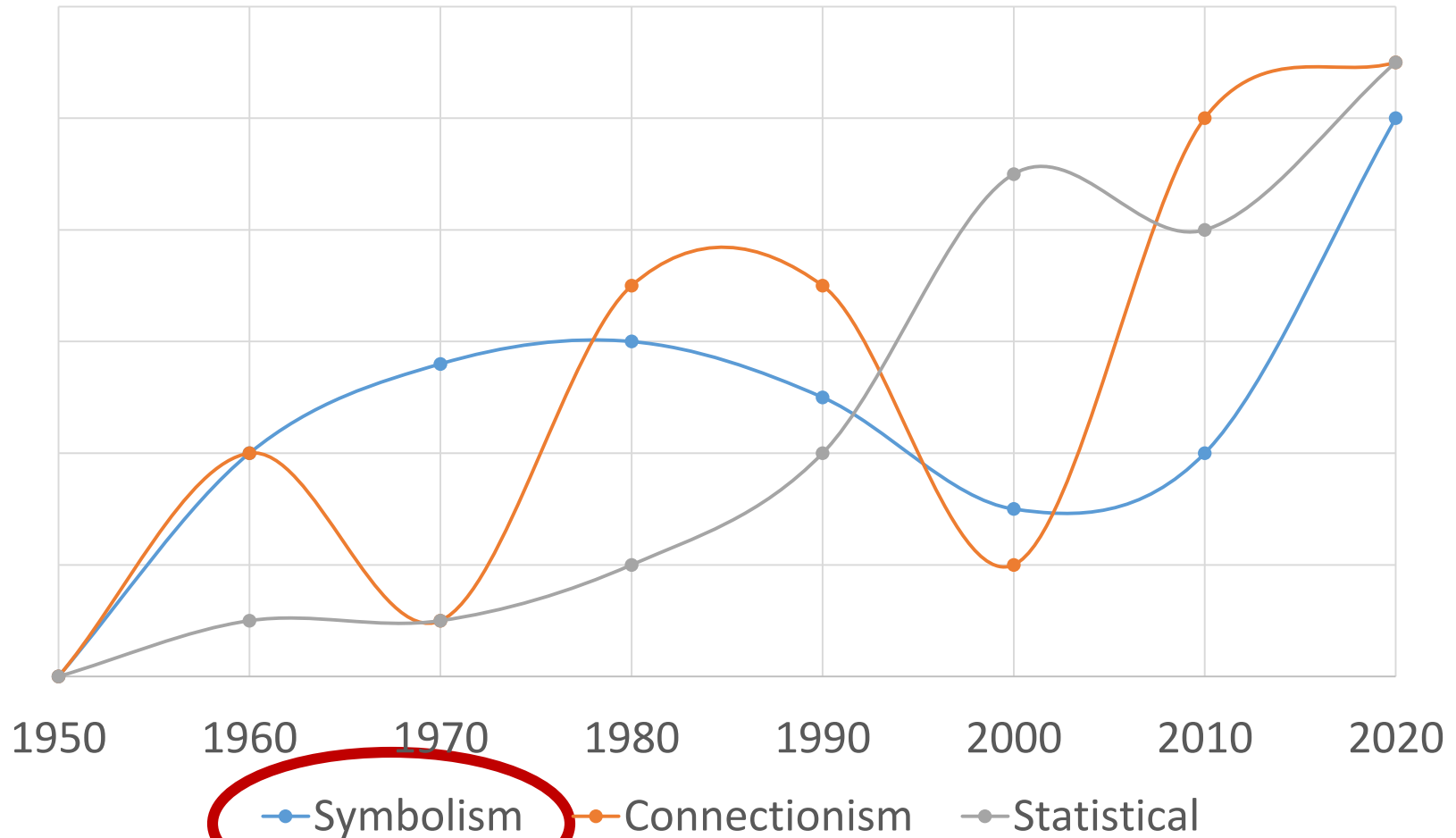


Three types of (strong) AI approaches



Propositional Logic

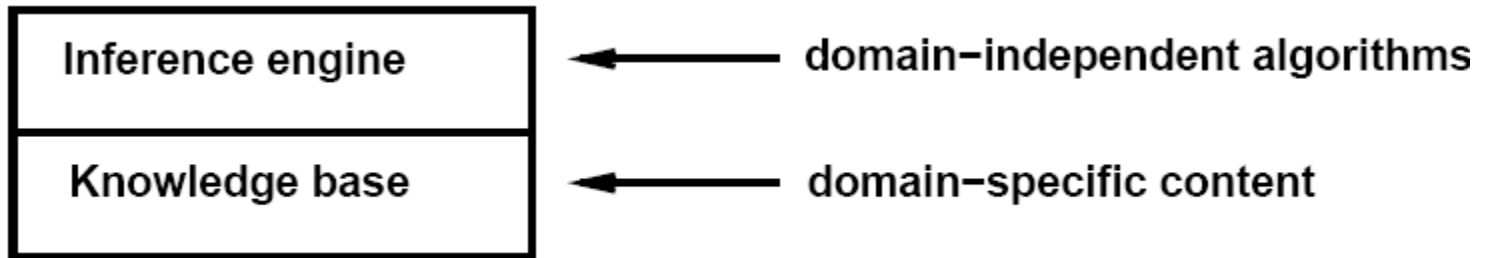
AIMA Chapter 7

Logic-based Symbolic AI

- Logic
 - Formal language in which knowledge can be expressed
 - A means of carrying out reasoning in the language

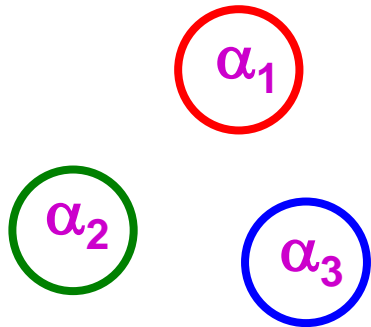
Logic-based Symbolic AI

- Logic (Knowledge-Based) AI
 - Knowledge base
 - set of sentences in a formal language to represent knowledge about the “world”
 - Inference engine
 - answers any answerable question following the knowledge base

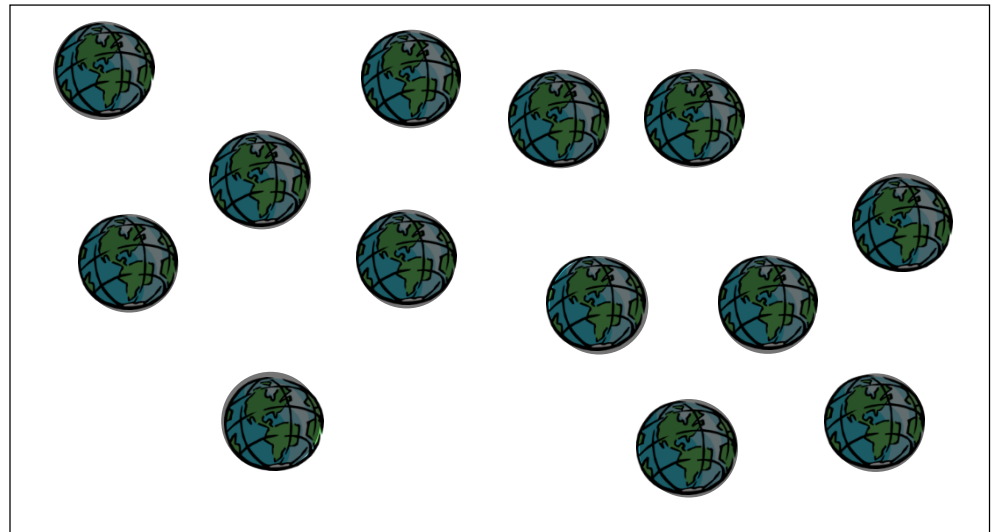


Formal Language

- Components of a formal language in a logic
 - **Syntax**: What sentences are allowed?
 - **Semantics**:
 - Which sentences are true/false in each **model** (possible world)?



Syntaxland



Semanticsland

Formal Language

- Example: the language of arithmetic
 - Syntax
 - $x+2 \geq y$ is a sentence
 - $x^2+y > \{\}$ is not a sentence
 - Semantics
 - $x+2 \geq y$ is true in a world where $x = 7, y = 1$
 - $x+2 \geq y$ is false in a world where $x = 0, y = 6$

Propositional Logic

Propositional logic: Syntax

- **Propositional logic** is the “simplest” logic
 - The proposition symbols $P1$, $P2$, etc. are sentences
 - If S is a sentence, $\neg S$ is a sentence (**negation**)
 - If $S1$ and $S2$ are sentences, $S1 \wedge S2$ is a sentence (**conjunction**)
 - If $S1$ and $S2$ are sentences, $S1 \vee S2$ is a sentence (**disjunction**)
 - If $S1$ and $S2$ are sentences, $S1 \Rightarrow S2$ is a sentence (**implication**)
 - If $S1$ and $S2$ are sentences, $S1 \Leftrightarrow S2$ is a sentence (**biconditional**)

\neg , \wedge , \vee , \Rightarrow , \Leftrightarrow are called logic connectives or operators



Sometimes \rightarrow and \leftrightarrow are used

Examples of PL sentences

- P means “It is hot.”
- Q means “It is humid.”
- R means “It is raining.”
- $(P \wedge Q) \Rightarrow R$
 - “If it is hot and humid, then it is raining”
- $Q \Rightarrow P$
 - “If it is humid, then it is hot”

Propositional logic: Semantics

- Each model specifies true/false for each proposition symbol

– E.g. P_1 P_2 P_3
 false true false

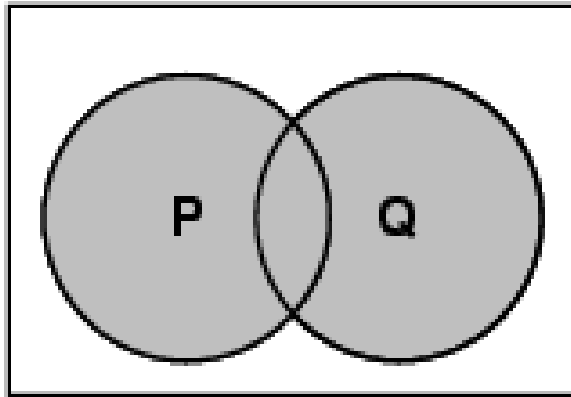
- Rules for evaluating truth with respect to a model m :
 - $\neg S$ is true iff S is false
 - $S_1 \wedge S_2$ is true iff S_1 is true and S_2 is true
 - $S_1 \vee S_2$ is true iff S_1 is true or S_2 is true
 - $S_1 \Rightarrow S_2$ is true iff S_1 is false or S_2 is true
 - $S_1 \Leftrightarrow S_2$ is true iff $S_1 \Rightarrow S_2$ is true and $S_2 \Rightarrow S_1$ is true

Truth tables for connectives

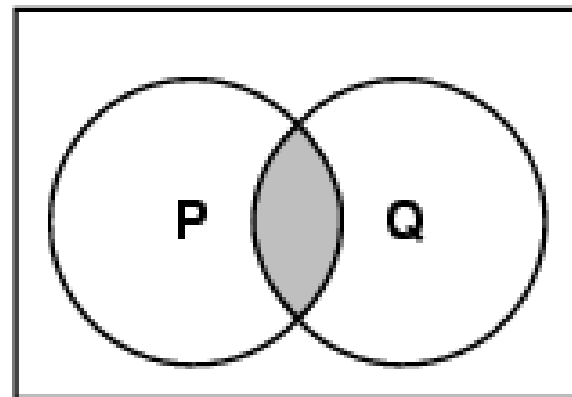
P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$	$P \Leftrightarrow Q$
<i>false</i>	<i>false</i>	<i>true</i>	<i>false</i>	<i>false</i>	<i>true</i>	<i>true</i>
<i>false</i>	<i>true</i>	<i>true</i>	<i>false</i>	<i>true</i>	<i>true</i>	<i>false</i>
<i>true</i>	<i>false</i>	<i>false</i>	<i>false</i>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

Venn Diagrams

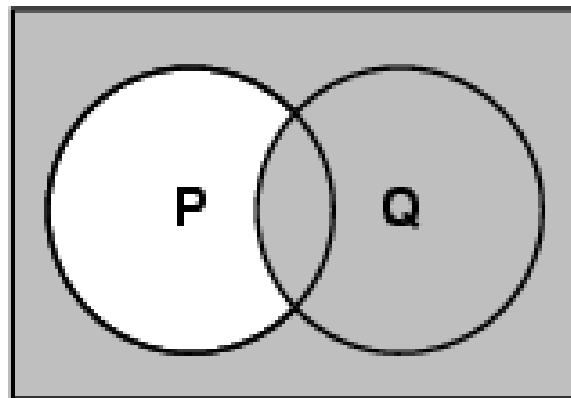
$$P \vee Q$$



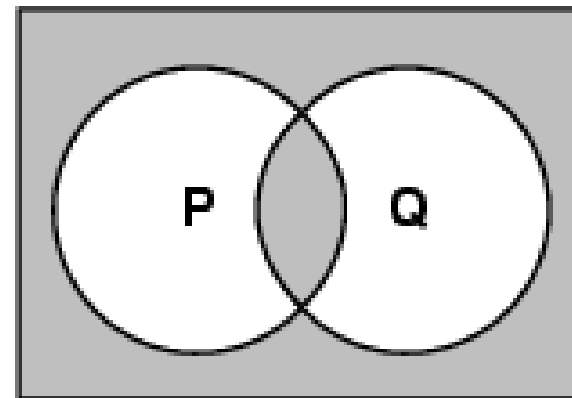
$$P \wedge Q$$



$$P \Rightarrow Q$$



$$P \Leftrightarrow Q$$



Material Implication

- $S1 \Rightarrow S2$ is true iff $S1$ is false or $S2$ is true
- Given the following propositions, is “ $S1 \Rightarrow S2$ ” true?
 - $S1$ means “the moon is made of green cheese”
 - $S2$ means “the world is coming to an end”
- Material implication does not capture the meaning of “if... then”.
- See “[Paradoxes of material implication](#)” in Wikipedia

Logical equivalence

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

$$(\alpha \wedge \beta) \equiv (\beta \wedge \alpha) \quad \text{commutativity of } \wedge$$

$$(\alpha \vee \beta) \equiv (\beta \vee \alpha) \quad \text{commutativity of } \vee$$

$$((\alpha \wedge \beta) \wedge \gamma) \equiv (\alpha \wedge (\beta \wedge \gamma)) \quad \text{associativity of } \wedge$$

$$((\alpha \vee \beta) \vee \gamma) \equiv (\alpha \vee (\beta \vee \gamma)) \quad \text{associativity of } \vee$$

$$\neg(\neg\alpha) \equiv \alpha \quad \text{double-negation elimination}$$

$$(\alpha \Rightarrow \beta) \equiv (\neg\beta \Rightarrow \neg\alpha) \quad \text{contraposition}$$

$$(\alpha \Rightarrow \beta) \equiv (\neg\alpha \vee \beta) \quad \text{implication elimination}$$

$$(\alpha \Leftrightarrow \beta) \equiv ((\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)) \quad \text{biconditional elimination}$$

$$\neg(\alpha \wedge \beta) \equiv (\neg\alpha \vee \neg\beta) \quad \text{de Morgan}$$

$$\neg(\alpha \vee \beta) \equiv (\neg\alpha \wedge \neg\beta) \quad \text{de Morgan}$$

$$(\alpha \wedge (\beta \vee \gamma)) \equiv ((\alpha \wedge \beta) \vee (\alpha \wedge \gamma)) \quad \text{distributivity of } \wedge \text{ over } \vee$$

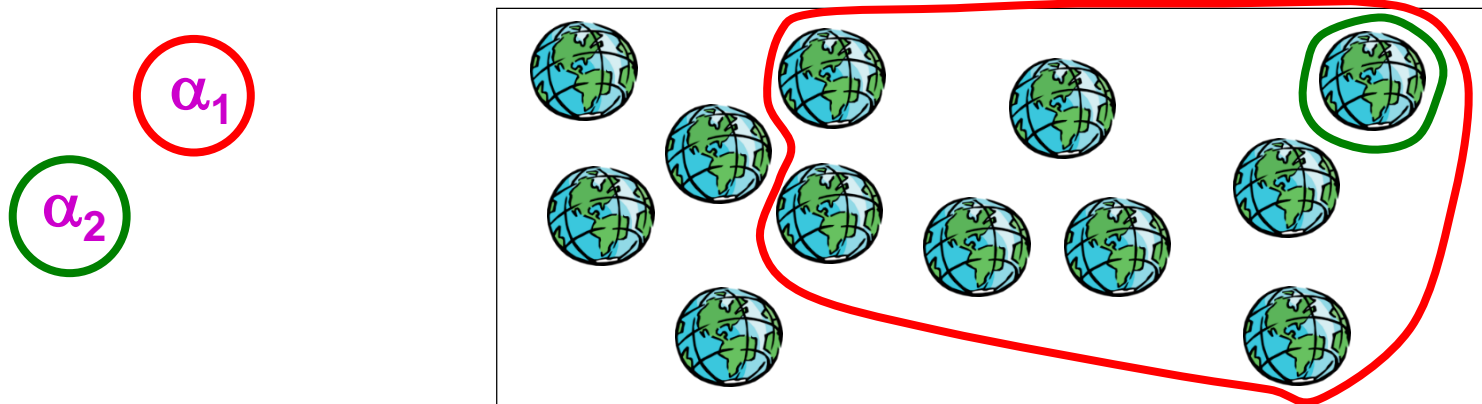
$$(\alpha \vee (\beta \wedge \gamma)) \equiv ((\alpha \vee \beta) \wedge (\alpha \vee \gamma)) \quad \text{distributivity of } \vee \text{ over } \wedge$$

Validity and satisfiability

- A sentence is **valid** if it is true in all models
 - e.g., $A \vee \neg A$, $A \Rightarrow A$, $(A \wedge (A \Rightarrow B)) \Rightarrow B$
- 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., $A \wedge \neg A$
- Obviously, S is valid iff. $\neg S$ is unsatisfiable

Inference: entailment

- **Entailment:** $\alpha \models \beta$ (“ α entails β ” or “ β follows from α ”) means in every world where α is true, β is also true
 - i.e., the α -worlds are a subset of the β -worlds [$\text{models}(\alpha) \subseteq \text{models}(\beta)$]
- In the example, $\alpha_2 \models \alpha_1$



Inference: proof

- A **proof** ($\alpha \models \beta$) is a demonstration of entailment from α to β
 - Method 1: model checking
 - Truth table enumeration to check if $\text{models}(\alpha) \subseteq \text{models}(\beta)$
 - Time complexity always exponential in n ☹

P1	P2	...	Pn	α	β
F	F	...	F	F	T
F	F	...	T	T	T
.....					
T	T	...	F	T	T
T	T	...	T	F	F

Inference: proof

- A **proof** ($\alpha \vdash \beta$) is a demonstration of entailment from α to β
 - Method 2: application of inference rules
 - Search for a finite sequence of sentences each of which is an **axiom** or follows from the preceding sentences by a **rule of inference**
 - Axiom: a sentence known to be true
 - Rule of inference: a function that takes one or more sentences (premises) and returns a sentence (conclusion)

Inference: soundness & completeness

- **Sound** inference
 - everything that can be proved is in fact entailed
- **Complete** inference
 - everything that is entailed can be proved
- Method 1 (enumeration) is obviously sound and complete
- For method 2 (applying inference rules), it is much less obvious
 - Example: arithmetic is found to be not complete! (Gödel's theorem, 1931)

Quiz

- What's the connection between complete inference algorithms and complete search algorithms?
- Answer 1: they both have the words “complete...algorithm”
- Answer 2: Formulate inference $\alpha \vdash \beta$ as a search problem
 - Initial state: KB contains α
 - Actions: apply any inference rule that matches KB, add conclusion
 - Goal test: KB contains β

Hence any complete search algorithm can be used to produce a complete inference algorithm

Resolution: an inference rule in PL

- **Conjunctive Normal Form (CNF)**
 - conjunction of disjunctions of literals (clauses)
 - Ex
 - $(A \vee \neg B) \wedge (B \vee \neg C \vee \neg D)$
 - $(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge (\neg P_{1,2} \vee B_{1,1}) \wedge (\neg P_{2,1} \vee B_{1,1})$

Conversion to CNF

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

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

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

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:

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

Resolution: an inference rule in PL

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

Suppose l_i is $\neg m_j$

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

Examples:

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

$$\frac{P_1, \neg P_1}{\{}}$$

- Resolution is sound and complete for propositional logic

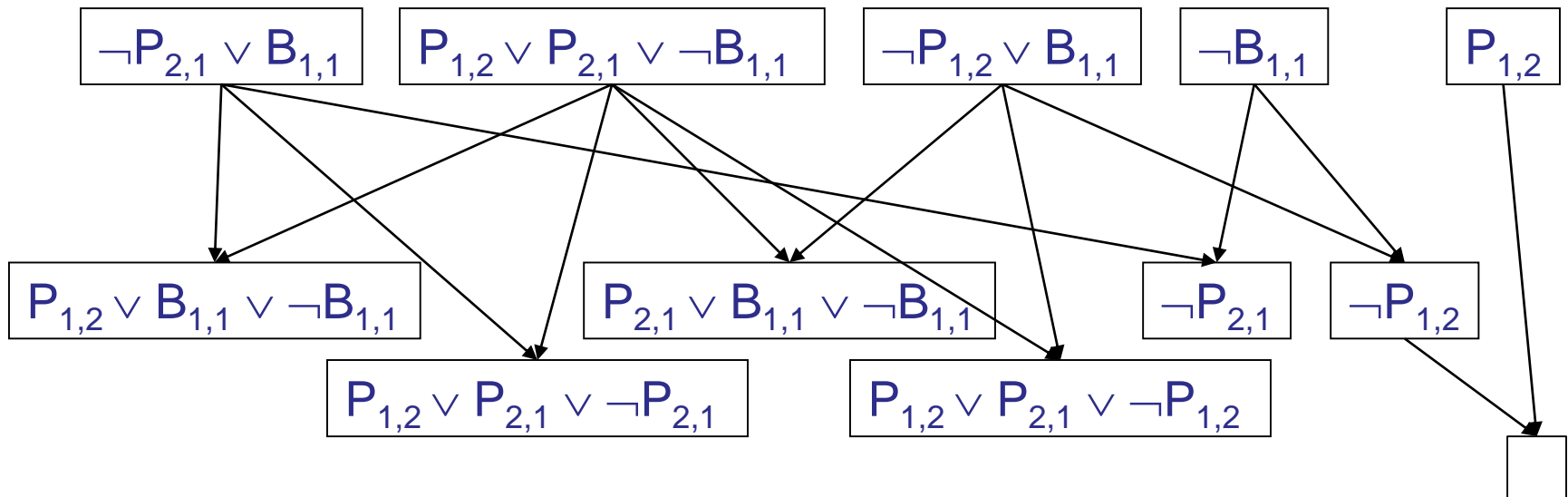
Resolution algorithm

- The best way to prove $KB \models \alpha$?
 - **Proof by contradiction**, i.e., show $KB \wedge \neg \alpha$ is unsatisfiable
- 1. Convert $KB \wedge \neg \alpha$ to CNF
- 2. Repeatedly apply the resolution rule to add new clauses, until one of the two things happens
 - a) Two clauses resolve to yield the empty clause, in which case KB entails α
 - b) There is no new clause that can be added, in which case KB does not entail α

Resolution example


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

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



Horn Logic

Horn logic

- Inference in propositional logic is in general NP-complete!
 - Solution: a **subset** of propositional logic that supports **efficient** inference
- 
- Expressiveness vs. Inference difficulty!!

- **Horn logic**: only (strict) **Horn clauses** are allowed
 - A Horn clause has the form:
$$P1 \wedge P2 \wedge P3 \dots \wedge Pn \Rightarrow Q$$
or alternatively
$$\neg P1 \vee \neg P2 \vee \neg P3 \dots \vee \neg Pn \vee Q$$
where Ps and Q are non-negated proposition symbols (atoms)
 - n can be zero, i.e., the clause contains a single atom

Inference in Horn logic

- **Modus Ponens**

$$\frac{\alpha_1, \dots, \alpha_n, \quad \overbrace{\alpha_1 \wedge \dots \wedge \alpha_n}^{\text{premises}} \Rightarrow \beta^{\text{conclusion}}}{\beta}$$

- Modus Ponens is sound and complete for Horn logic
- Inference algorithms (for Horn logic)
 - Forward chaining, backward chaining
 - These algorithms are very natural and run in linear time

Forward chaining

- Idea: to prove $KB \models Q$
 - Add new clauses into the KB by applying Modus Ponens, until Q is added

$$P \Rightarrow Q$$

$$L \wedge M \Rightarrow P$$

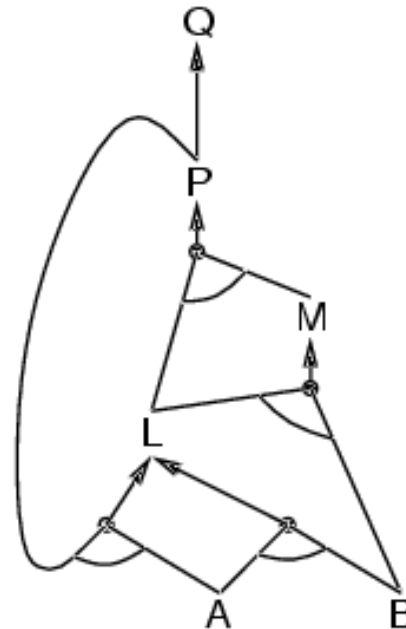
$$B \wedge L \Rightarrow M$$

$$A \wedge P \Rightarrow L$$

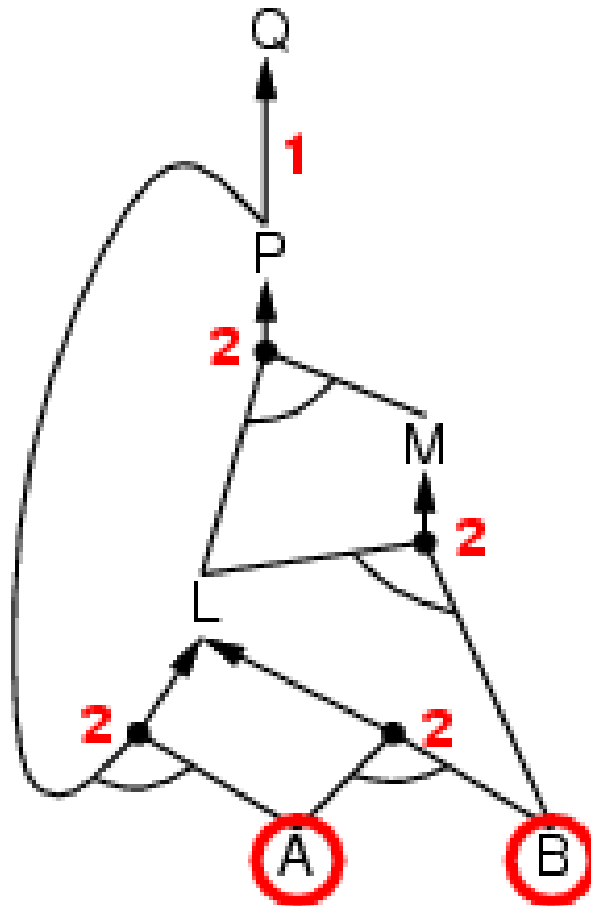
$$A \wedge B \Rightarrow L$$

A

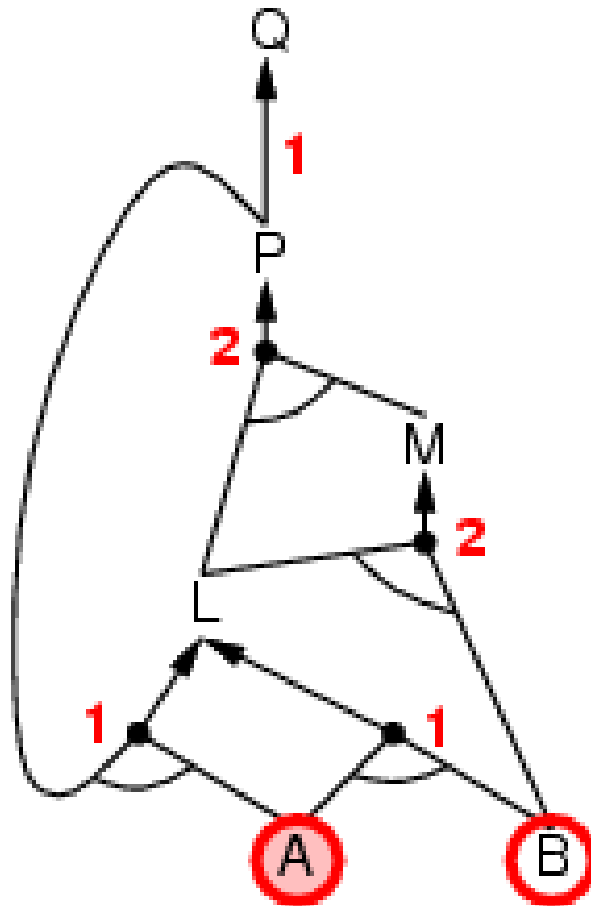
B



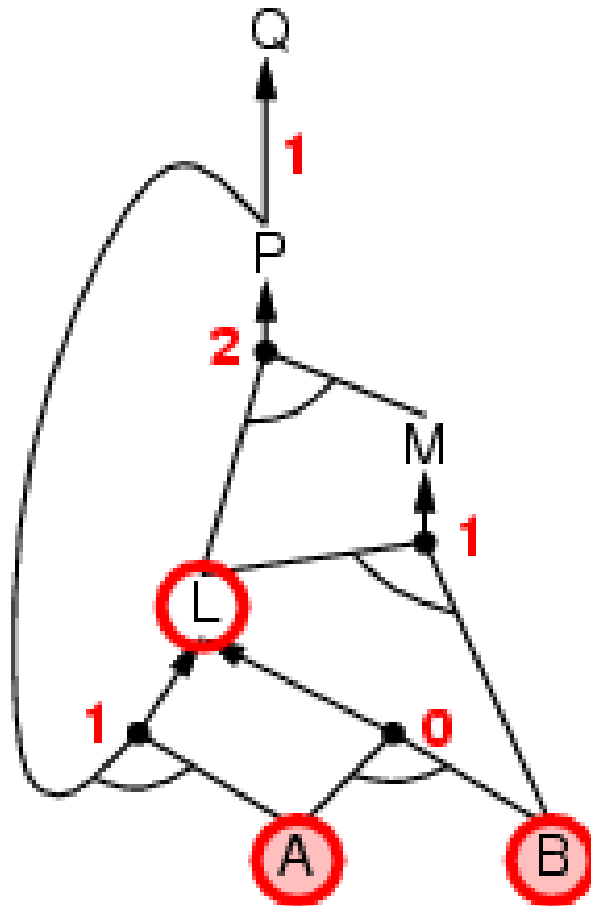
Forward chaining example



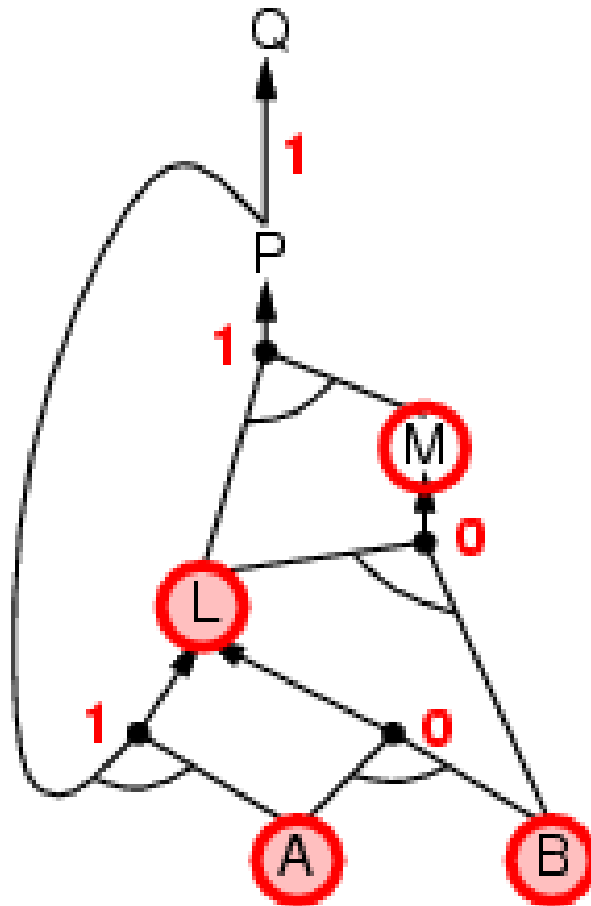
Forward chaining example



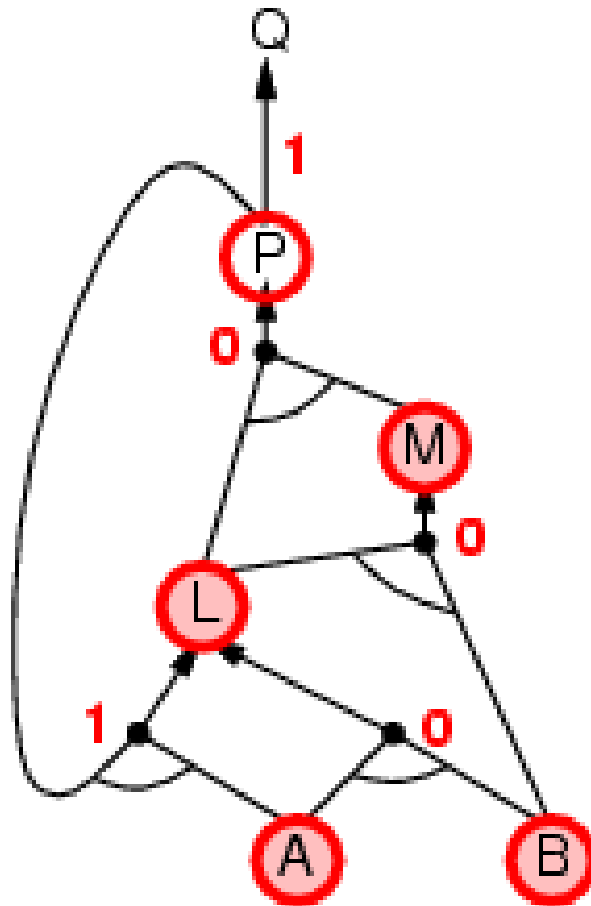
Forward chaining example



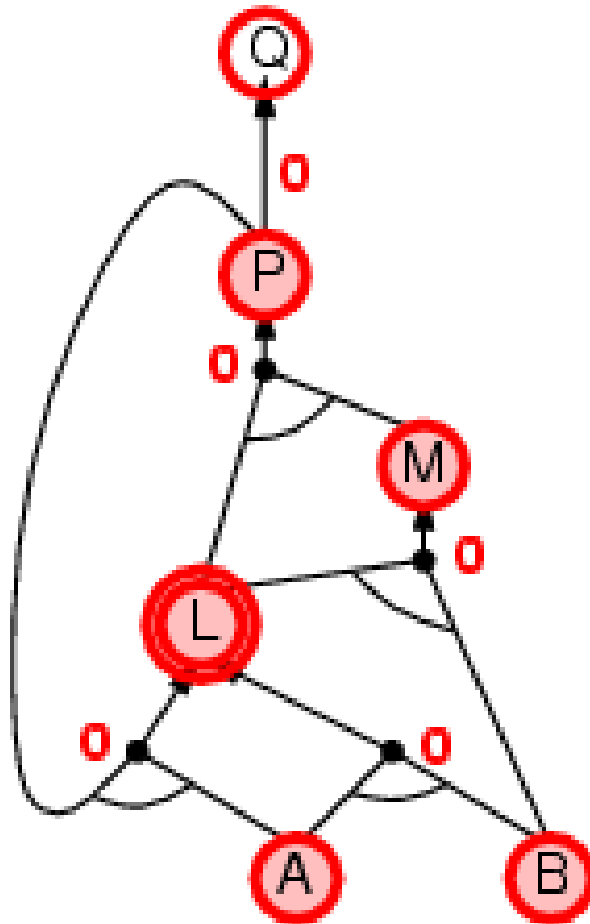
Forward chaining example



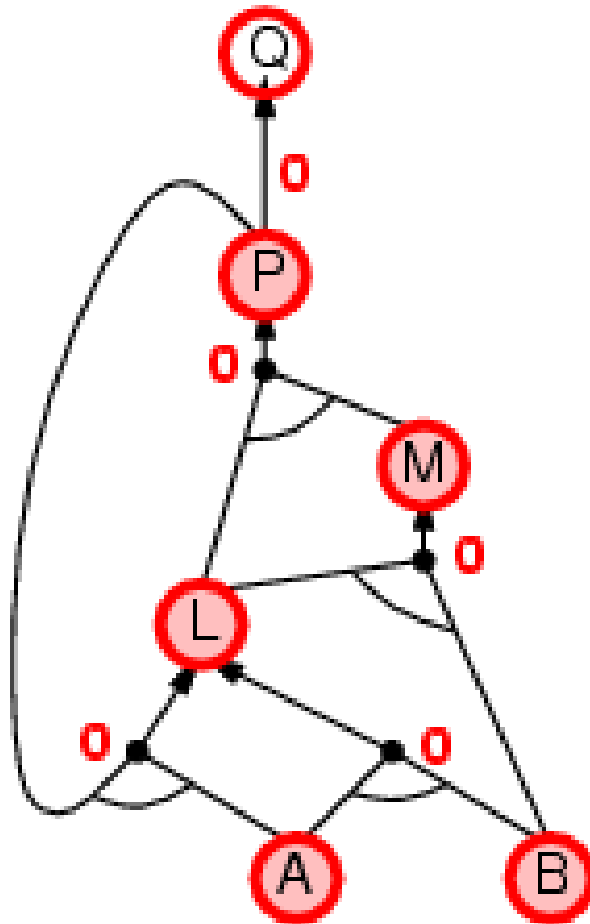
Forward chaining example



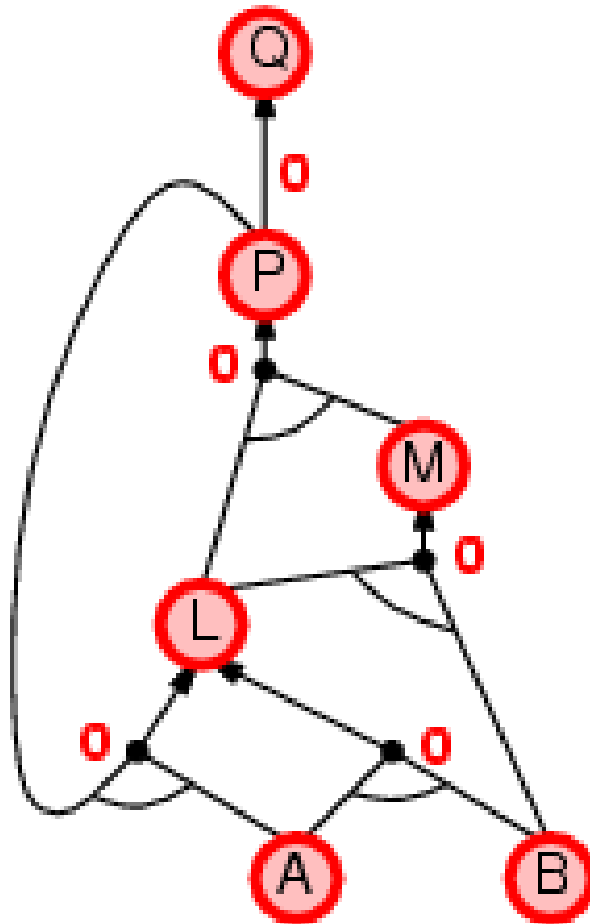
Forward chaining example



Forward chaining example



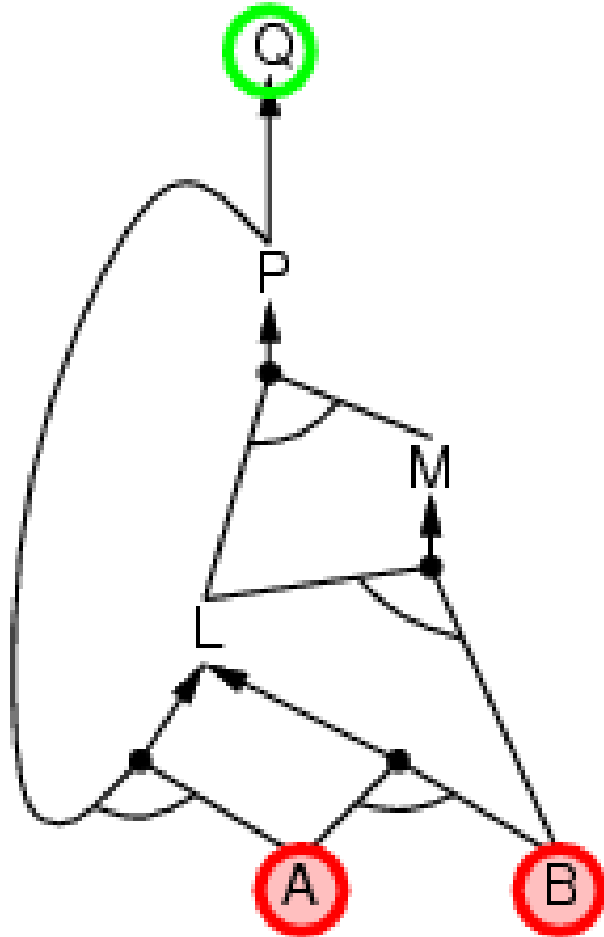
Forward chaining example



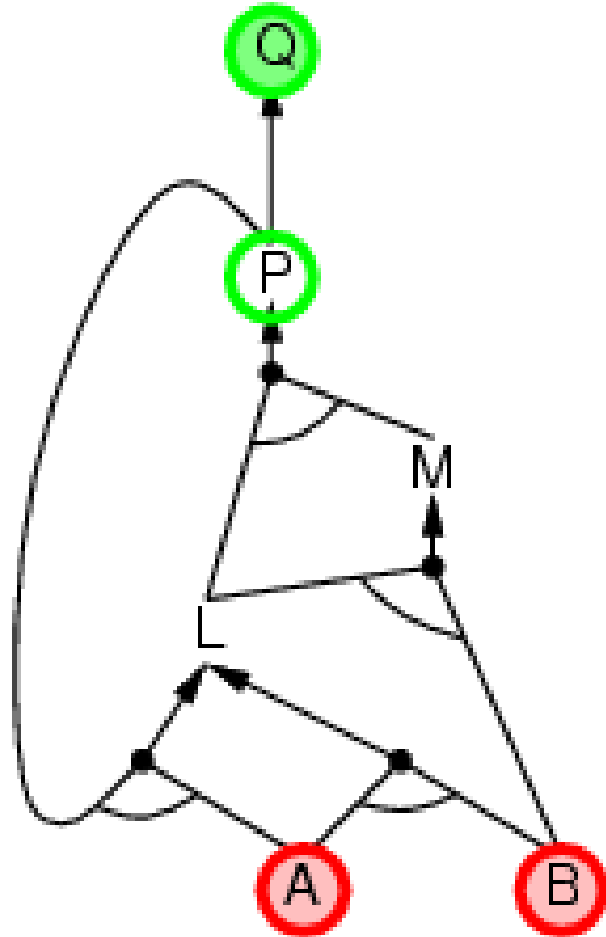
Backward chaining

- Idea: work backwards from the query q :
 - to prove q by BC,
 - check if q is known to be true already, or
 - prove by BC all premises of some rule concluding q
- Avoid loops: check if new subgoal is already on the goal stack
- Avoid repeated work: check if new subgoal
 1. has already been proved true, or
 2. has already failed

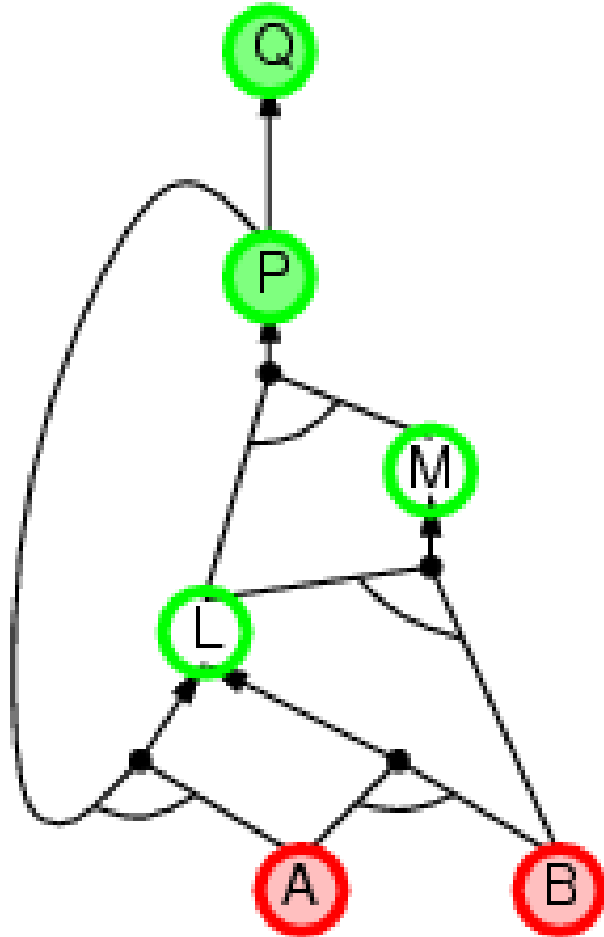
Backward chaining example



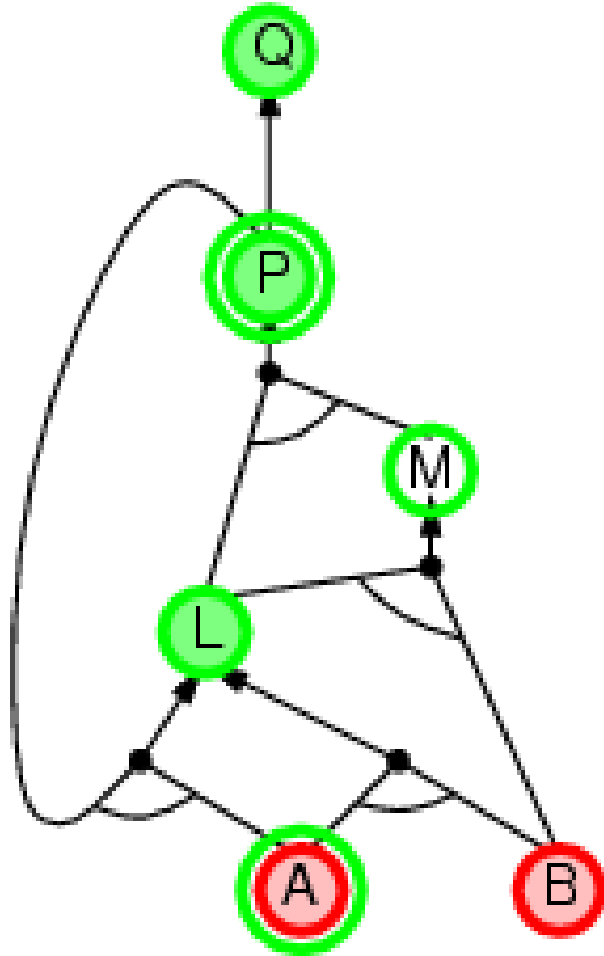
Backward chaining example



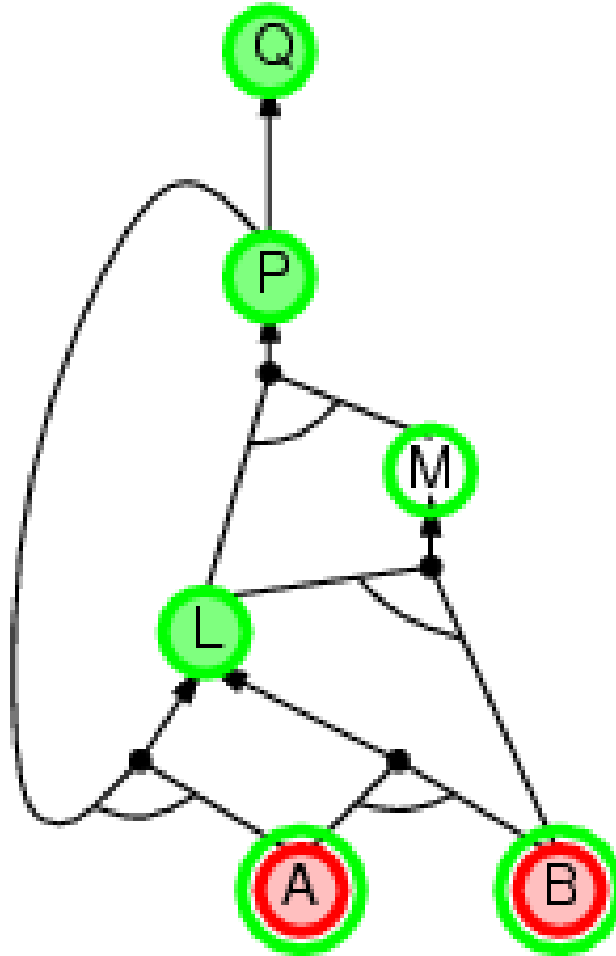
Backward chaining example



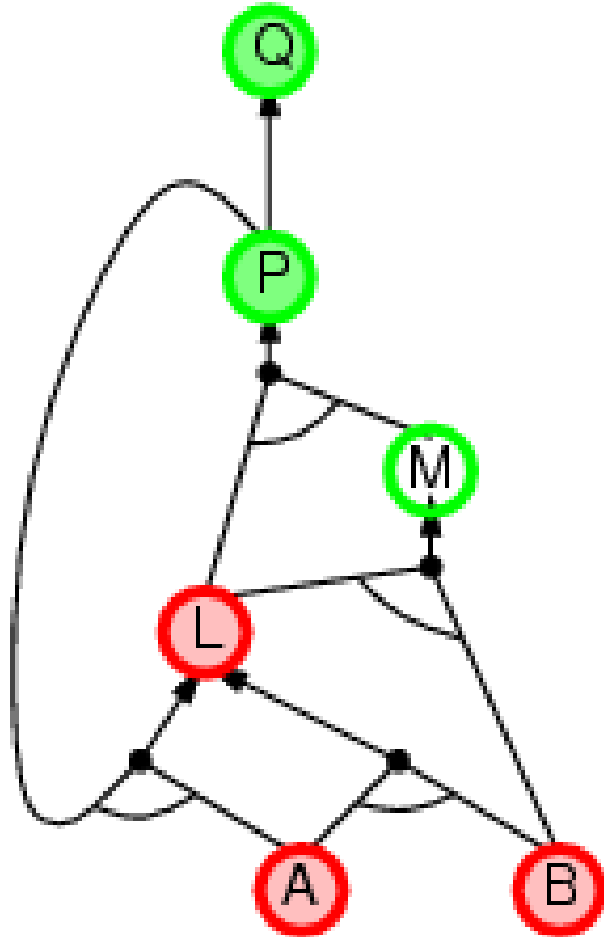
Backward chaining example



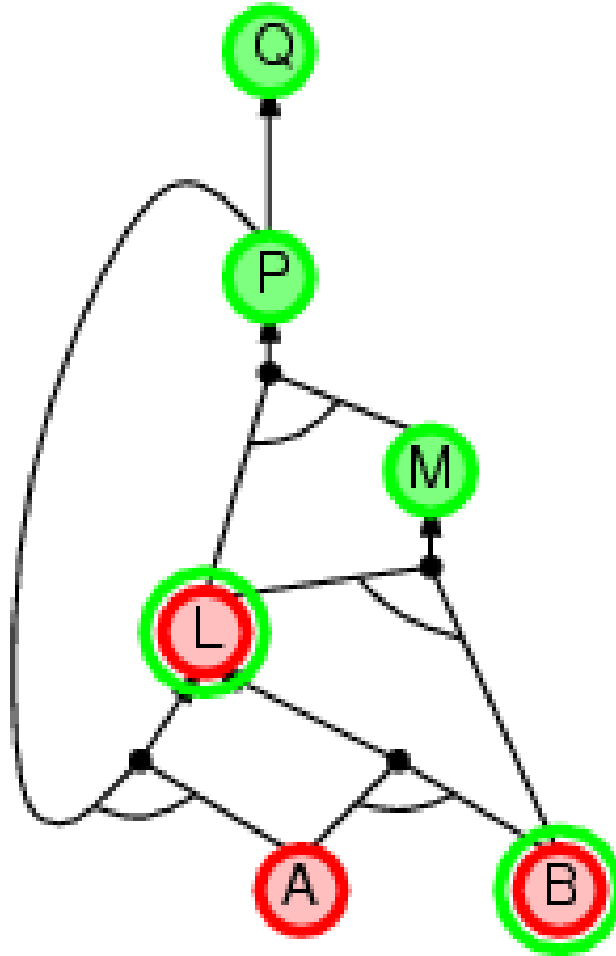
Backward chaining example



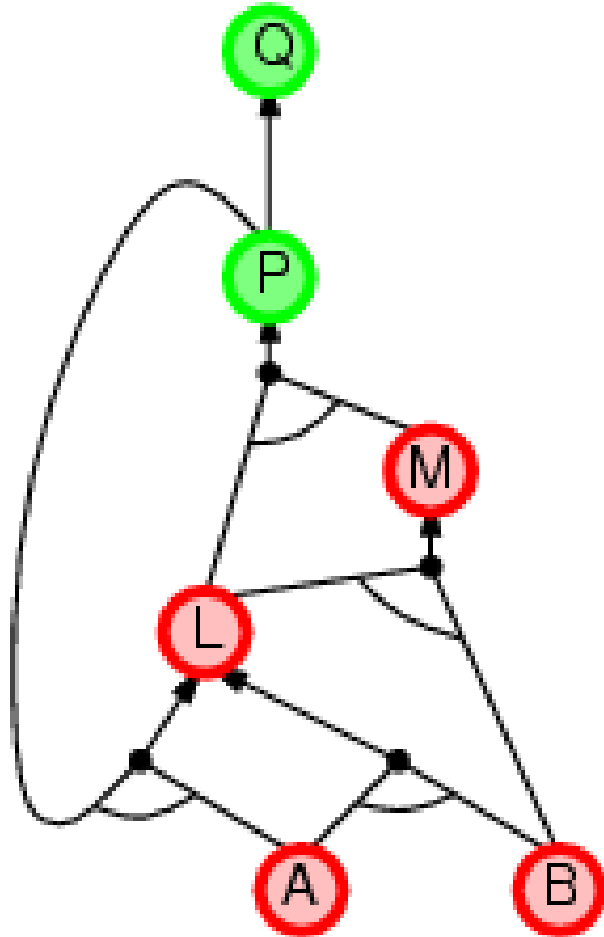
Backward chaining example



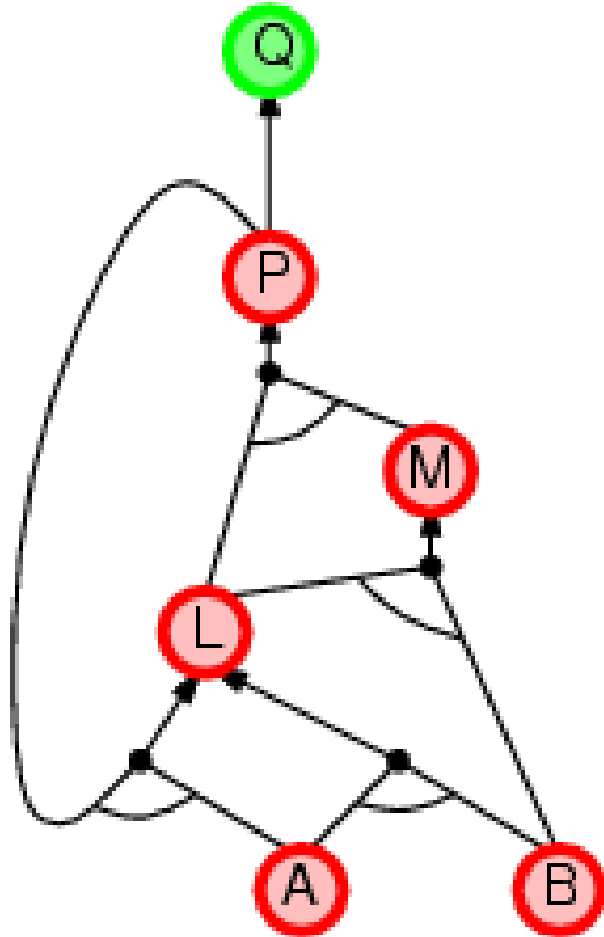
Backward chaining example



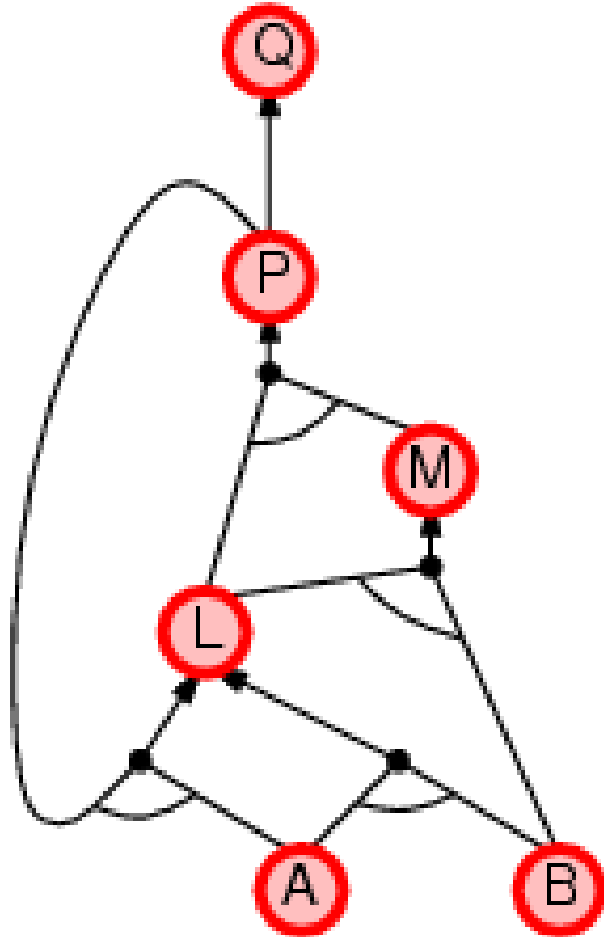
Backward chaining example



Backward chaining example



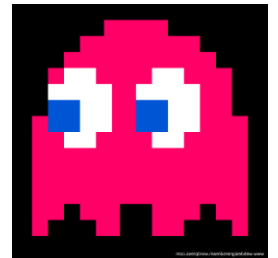
Backward chaining example



Forward vs. backward chaining

- 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

A Logical Pacman



Partially observable Pacman

- Pacman perceives just the local walls/gaps
- Formulation: what proposition symbols do we need?
 - Pacman's location
 - $At_{1,1}_0$ (Pacman is at [1,1] at time 0) $At_{3,3}_1$ etc
 - Wall locations
 - $Wall_{0,0}$ $Wall_{0,1}$ etc
 - Percepts
 - $Blocked_W_0$ (blocked by wall to my West at time 0) etc.
 - Actions
 - W_0 (Pacman moves West at time 0) E_0 etc.
- $N \times N$ world for T time steps $\Rightarrow N^2T + N^2 + 4T + 4T = O(N^2T)$ symbols

Sensor model

- State facts about how Pacman's percepts arise...
- Pacman perceives a wall to the West at time t if he is in x,y and there is a wall at $x-1,y$

$((At_1,1_0 \wedge Wall_0,1) \vee$

$(At_1,2_0 \wedge Wall_0,2) \vee$

$(At_1,3_0 \wedge Wall_0,3) \vee \dots) \Rightarrow Blocked_W_0$

Sensor model

- State facts about how Pacman's percepts arise...
- Pacman perceives a wall to the West at time t *if and only if* he is in x,y and there is a wall at x-1,y

$$\begin{aligned} \text{Blocked_W_0} \Leftrightarrow & ((\text{At_1,1_0} \wedge \text{Wall_0,1}) \vee \\ & (\text{At_1,2_0} \wedge \text{Wall_0,2}) \vee \\ & (\text{At_1,3_0} \wedge \text{Wall_0,3}) \vee \dots) \end{aligned}$$

Transition model

- How does each state symbol at each time get its value?
 - E.g., should $At_{3,3}_{17}$ be T or F?
- A state symbol gets its value according to a successor-state axiom

$$X_t \Leftrightarrow [X_{t-1} \wedge \neg(\text{some action}_{t-1} \text{ made it false})] \vee [\neg X_{t-1} \wedge (\text{some action}_{t-1} \text{ made it true})]$$

- For Pacman location:

$$\begin{aligned} At_{3,3}_{17} \Leftrightarrow & [At_{3,3}_{16} \wedge \neg((\neg Wall_{3,4} \wedge N_{16}) \vee (\neg Wall_{4,3} \wedge E_{16}) \vee \dots)] \\ & \vee [\neg At_{3,3}_{16} \wedge ((At_{3,2}_{16} \wedge \neg Wall_{3,3} \wedge N_{16}) \vee \\ & (At_{2,3}_{16} \wedge \neg Wall_{3,3} \wedge E_{16}) \vee \dots)] \end{aligned}$$

Initial state

- The agent may know its initial location:

$At_1,1_0$

- Or, it may not:

$At_1,1_0 \vee At_1,2_0 \vee At_1,3_0 \vee \dots \vee At_3,3_0$

Domain constraint

- Pacman cannot be in two places at once!
 $\neg(\text{At_1,1_0} \wedge \text{At_1,2_0}) \wedge \neg(\text{At_1,1_0} \wedge \text{At_1,3_0}) \wedge \dots$
 $\neg(\text{At_1,1_1} \wedge \text{At_1,2_1}) \wedge \neg(\text{At_1,1_1} \wedge \text{At_1,3_1}) \wedge \dots$
...
- Pacman cannot take two actions at the same time!
 $\neg(\text{E_0} \wedge \text{S_0}) \wedge \neg(\text{E_0} \wedge \text{W_0}) \wedge \dots$
 $\neg(\text{E_1} \wedge \text{S_1}) \wedge \neg(\text{E_1} \wedge \text{W_1}) \wedge \dots$
...
- Pacman cannot go into a wall!
 $\text{At_1,1_0} \wedge \text{N_0} \Rightarrow \neg\text{Wall_1,2}$
...

Planning as satisfiability

- SAT solver
 - Input: a logic expression
 - Output: a model (true/false assignments to symbols) that satisfies the expression if such a model exists
- Can we use it to make plans for Pacman (e.g., to move to a goal position)?
 - For $T = 1$ to infinity, set up the KB as follows and run SAT solver:
 - Initial state, domain constraints, sensor & transition model sentences up to time T
 - Goal is true at time T
 - If a model is returned, extract a plan from action assignment

Planning as satisfiability

- Q: Isn't this a search problem? Any advantage of using logic?
- A: We can use logic to solve not only search problems, but any problems that are representable using the logic.

Logic programming

- Ordinary programming
 - Identify problem
 - Assemble information
 - Figure out solution
 - Encode solution
 - Encode problem instance as data
 - Apply program to data
- Logic programming
 - Identify problem
 - Assemble information
 - **<coffee break>** 😊
 - Encode info in KB
 - Encode problem instance as facts
 - Ask queries (run SAT solver)

Summary

- Logic
 - Logical AI applies **inference** to a **knowledge base** to derive new information
- Propositional logic
 - Syntax
 - Semantics
 - Inference (resolution)
- Horn logic
 - Inference (forward/backward chaining)
- Application of logic to Pacman