

CS:4420 Artificial Intelligence

Spring 2017

Propositional Logic

Cesare Tinelli

The University of Iowa

Copyright 2004–17, Cesare Tinelli and Stuart Russell ^a

^a These notes were originally developed by Stuart Russell and are used with permission. They are copyrighted material and may not be used in other course settings outside of the University of Iowa in their current or modified form without the express written consent of the copyright holders.

Readings

- Chap. 7 of [Russell and Norvig, 2012]

Logics

A **logic** is a triple $\langle \mathcal{L}, \mathcal{S}, \mathcal{R} \rangle$ where

- \mathcal{L} , the logic's **language**, is a class of sentences described by a formal grammar
- \mathcal{S} , the logic's **semantics** is a formal specification of how to assign *meaning* in the “real world” to the elements of \mathcal{L}
- \mathcal{R} , the logic's **inference system**, is a set of formal derivation *rules* over \mathcal{L}

There are **several** logics: propositional, first-order, higher-order, modal, temporal, intuitionistic, linear, equational, non-monotonic, fuzzy, ...

We will concentrate on **propositional logic** and **first-order logic**

Propositional Logic

Each sentence is made of

- propositional variables (A, B, \dots, P, Q, \dots)
- logical constants (**True, False**)
- logical connectives ($\wedge, \vee, \Rightarrow, \dots$)

Every propositional variable stands for a basic **fact**

Examples: *I'm hungry, Apples are red, Joe and Jill are married*

Propositional Logic

Ontological Commitments

Propositional Logic is about **facts** in the world that are either true or false, nothing else

Semantics of Propositional Logic

Since each propositional variable stands for a fact about the world, its meaning ranges over the Boolean values $\{true, false\}$

Note: Do not confuse

- *true, false*, which are values (i.e., semantical entities) here with
- **True, False**, which are logical constants (i.e., symbols of the language)

Propositional Logic

The Language

- Each propositional variable (A, B, \dots, P, Q, \dots) is a sentence
- Each logical constant (**True, False**) is a sentence
- If φ and ψ are sentences, all of the following are also sentences

(φ) $\neg\varphi$ $\varphi \wedge \psi$ $\varphi \vee \psi$ $\varphi \Rightarrow \psi$ $\varphi \Leftrightarrow \psi$

- Nothing else is a sentence

The Language of Propositional Logic

More formally, it is the language generated by the following grammar

Symbols:

- Propositional variables: A, B, \dots, P, Q, \dots
- Logical constants:

True	(true)	\wedge	(and)	\Rightarrow	(implies)	\neg	(not)
False	(false)	\vee	(or)	\Leftrightarrow	(equivalent)		

Grammar Rules:

Sentence ::= *AtomicS* | *ComplexS*

AtomicS ::= **True** | **False** | A | B | ... | P | Q | ...

ComplexS ::= (*Sentence*) | *Sentence* *Connective* *Sentence* | \neg *Sentence*

Connective ::= \wedge | \vee | \Rightarrow | \Leftrightarrow

Wumpus world sentences

Let $P_{i,j}$ be true if there is a pit in $[i, j]$

Let $B_{i,j}$ be true if there is a breeze in $[i, j]$

“Pits cause breezes in adjacent squares”

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

$$\neg A$$

$$B_{2,1}$$

Wumpus world sentences

Let $P_{i,j}$ be true if there is a pit in $[i, j]$

Let $B_{i,j}$ be true if there is a breeze in $[i, j]$

“Pits cause breezes in adjacent squares”

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

$$\neg A$$

$$B_{2,1}$$

“A square is breezy **if and only if** there is an adjacent pit”

$$A \Leftrightarrow (B \vee C)$$

$$B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$$

Semantics of Propositional Logic

The meaning of **True** is always *true*

The meaning of **False** is always *false*

The meaning of the other sentences depends on the meaning of the propositional variables

- **Base cases:** truth tables

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>

- **Non-base Cases:** given by reduction to the base cases

Ex: the meaning of $(P \vee Q) \wedge R$ is the same as the meaning of $A \wedge R$ where A has the same meaning as $P \vee Q$

Semantics of Propositional Logic

An assignment of Boolean values to the propositional variables of a sentence is an **interpretation** of the sentence

P	H	$P \vee H$	$(P \vee H) \wedge \neg H$	$((P \vee H) \wedge \neg H) \Rightarrow P$
<i>False</i>	<i>False</i>	<i>False</i>	<i>False</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>True</i>	<i>True</i>
<i>True</i>	<i>True</i>	<i>True</i>	<i>False</i>	<i>True</i>

Interpretations: $\{P \mapsto \text{false}, H \mapsto \text{false}\}, \{P \mapsto \text{false}, H \mapsto \text{true}\}, \dots$

The semantics of Propositional Logic is **compositional**:
the meaning of a sentence is defined recursively in terms of the
meaning of the sentence's components

Semantics of Propositional Logic

The meaning of a sentence in general depends on its interpretation
Some sentences, however, have always the same meaning

P	H	$P \vee H$	$(P \vee H) \wedge \neg H$	$((P \vee H) \wedge \neg H) \Rightarrow P$
<i>False</i>	<i>False</i>	<i>False</i>	<i>False</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>True</i>	<i>True</i>
<i>True</i>	<i>True</i>	<i>True</i>	<i>False</i>	<i>True</i>

A sentence is

- **satisfiable** if it is true in **some** interpretation
- **unsatisfiable** if it is true in **no** interpretation
- **valid** if it is true in **every** possible interpretation
- **invalid** if it is false in **some** possible interpretation

A Warning

Disjunction

- $A \vee B$ is true when A or B or *or both* are true (inclusive or)
- $A \oplus B$ is sometimes used to mean “either A or B but not both” (exclusive or)

Implication

- $A \Rightarrow B$ does not require a causal connection between A and B
Ex: *Sky-is-blue* \Rightarrow *Snow-is-white*
- When A is false, $A \Rightarrow B$ is always true *regardless* of B
Ex: *Two-equals-four* \Rightarrow *Apples-are-red*
- Beware of negations in implications
Ex: *Is-a-female-bird* \Rightarrow *Lays-eggs*
 \neg *Is-a-female-bird* \Rightarrow \neg *lays-eggs*

Entailment in Propositional Logic

Given

- a set Γ of sentences and
- a sentence φ ,

we write

$$\Gamma \models \varphi$$

iff every interpretation that makes all sentences in Γ true makes φ also true

$\Gamma \models \varphi$ is read as “ Γ entails φ ” or “ φ logically follows from Γ ”

Entailment in Propositional Logic

Examples

$$\{A, A \Rightarrow B\} \models B$$

$$\{A\} \models A \vee B$$

$$\{A, B\} \models A \wedge B$$

$$\{\} \models A \vee \neg A$$

$$\{A\} \not\models A \wedge B$$

$$\{A \vee \neg A\} \not\models A$$

	A	B	$A \Rightarrow B$	$A \vee B$	$A \wedge B$	$A \vee \neg A$
1.	<i>false</i>	<i>false</i>	<i>true</i>	<i>false</i>	<i>false</i>	<i>true</i>
2.	<i>false</i>	<i>true</i>	<i>true</i>	<i>true</i>	<i>false</i>	<i>true</i>
3.	<i>true</i>	<i>false</i>	<i>false</i>	<i>true</i>	<i>false</i>	<i>true</i>
4.	<i>true</i>	<i>true</i>	<i>true</i>	<i>true</i>	<i>true</i>	<i>true</i>

Properties of Entailment

- $\Gamma \models \varphi$, for all $\varphi \in \Gamma$ (inclusion property of PL)
- if $\Gamma \models \varphi$, then $\Gamma' \models \varphi$ for all $\Gamma' \supseteq \Gamma$ (monotonicity of PL)
- φ is valid iff $\{\} \models \varphi$ (also written as $\models \varphi$)
- φ is unsatisfiable iff $\varphi \models \mathbf{False}$
- $\Gamma \models \varphi$ iff the set $\Gamma \cup \{\neg\varphi\}$ is unsatisfiable

Logical Equivalence

Two sentences φ_1 and φ_2 are **logically equivalent**, written

$$\varphi_1 \equiv \varphi_2$$

if $\varphi_1 \models \varphi_2$ and $\varphi_2 \models \varphi_1$

Note:

- $\varphi_1 \equiv \varphi_2$ if and only if every interpretation assigns the same Boolean value to φ_1 and φ_2
- Implication and equivalence (\Rightarrow , \Leftrightarrow), which are **syntactical entities**, are intimately related to entailment and logical equivalence (\models , \equiv), which are **semantical notions**:

$$\varphi_1 \models \varphi_2 \quad \text{iff} \quad \models \varphi_1 \Rightarrow \varphi_2$$

$$\varphi_1 \equiv \varphi_2 \quad \text{iff} \quad \models \varphi_1 \Leftrightarrow \varphi_2$$

Properties of Logical Connectives

- \wedge and \vee are **commutative**

$$\varphi_1 \wedge \varphi_2 \equiv \varphi_2 \wedge \varphi_1$$

$$\varphi_1 \vee \varphi_2 \equiv \varphi_2 \vee \varphi_1$$

- \wedge and \vee are *associative*

$$\varphi_1 \wedge (\varphi_2 \wedge \varphi_3) \equiv (\varphi_1 \wedge \varphi_2) \wedge \varphi_3$$

$$\varphi_1 \vee (\varphi_2 \vee \varphi_3) \equiv (\varphi_1 \vee \varphi_2) \vee \varphi_3$$

- \wedge and \vee are mutually *distributive*

$$\varphi_1 \wedge (\varphi_2 \vee \varphi_3) \equiv (\varphi_1 \wedge \varphi_2) \vee (\varphi_1 \wedge \varphi_3)$$

$$\varphi_1 \vee (\varphi_2 \wedge \varphi_3) \equiv (\varphi_1 \vee \varphi_2) \wedge (\varphi_1 \vee \varphi_3)$$

- \wedge and \vee are related by \neg (DeMorgan's Laws)

$$\neg(\varphi_1 \wedge \varphi_2) \equiv \neg\varphi_1 \vee \neg\varphi_2$$

$$\neg(\varphi_1 \vee \varphi_2) \equiv \neg\varphi_1 \wedge \neg\varphi_2$$

Properties of Logical Connectives

\wedge , \Rightarrow , and \Leftrightarrow are actually redundant:

$$\varphi_1 \wedge \varphi_2 \equiv \neg(\neg\varphi_1 \vee \neg\varphi_2)$$

$$\varphi_1 \Rightarrow \varphi_2 \equiv \neg\varphi_1 \vee \varphi_2$$

$$\varphi_1 \Leftrightarrow \varphi_2 \equiv (\varphi_1 \Rightarrow \varphi_2) \wedge (\varphi_2 \Rightarrow \varphi_1)$$

We keep them all mainly for convenience

Exercise Use the truth tables to verify all the logical equivalences seen so far

Inference Systems for Propositional Logic

An **inference system** \mathcal{I} for PL is a procedure that given a set $\Gamma = \{\alpha_1, \dots, \alpha_m\}$ of sentences and a sentence φ , may reply “yes”, “no”, or run forever

If \mathcal{I} replies positively to input (Γ, φ) , we say that Γ **derives** φ **in** \mathcal{I} (or, \mathcal{I} derives φ from Γ , or, φ derives from Γ in \mathcal{I}) and write

$$\Gamma \vdash_{\mathcal{I}} \varphi$$

Intuitively, \mathcal{I} should be such that it replies “yes” on input (Γ, φ) only if φ is in fact entailed by Γ

All These Fancy Symbols!

Note:

$A \wedge B \Rightarrow C$ is a sentence, a bunch of **symbols** manipulated by an **inference system \mathcal{I}**

$A \wedge B \models C$ is a mathematical abbreviation standing for the statement: “every interpretation that makes $A \wedge B$ true, makes C also true”

$A \wedge B \vdash_{\mathcal{I}} C$ is a mathematical abbreviation standing for the statement: “ \mathcal{I} derives C from $A \wedge B$ ”

In other words,

\Rightarrow is a formal symbol of the logic, which is used by the inference system

\models is a shorthand we use to talk about the meaning of formal sentences

$\vdash_{\mathcal{I}}$ is a shorthand we use to talk about the output of the inference system \mathcal{I}

All These Fancy Symbols!

The connective \Rightarrow and the shorthand \models are related as follows

The sentence $\varphi_1 \Rightarrow \varphi_2$ is valid (always true) if and only if $\varphi_1 \models \varphi_2$

Example: $A \Rightarrow (A \vee B)$ is valid and $A \models (A \vee B)$

	A	B	$A \vee B$	$A \Rightarrow (A \vee B)$
1.	<i>false</i>	<i>false</i>	<i>false</i>	<i>true</i>
2.	<i>false</i>	<i>true</i>	<i>true</i>	<i>true</i>
3.	<u><i>true</i></u>	<i>false</i>	<i>true</i>	<i>true</i>
4.	<u><i>true</i></u>	<i>true</i>	<i>true</i>	<i>true</i>

All These Fancy Symbols!

The shorthands \models and $\vdash_{\mathcal{I}}$ are related as follows

- A **sound** inference system can derive **only** sentences that logically follow from a given set of sentences:

$$\text{if } \Gamma \vdash_{\mathcal{I}} \varphi \text{ then } \Gamma \models \varphi$$

- A **complete** inference system can derive **all** sentences that logically follow from a given set of sentences:

$$\text{if } \Gamma \models \varphi \text{ then } \Gamma \vdash_{\mathcal{I}} \varphi$$

Inference systems for PL

Divided into (roughly) two kinds:

Rule-based

- Sound generation of new sentences from old
- **Proof** = a sequence of inference rule applications
Can use inference rules as operators as in a standard search procedures
- Typically require translation of sentences into some **normal form**

Model-based

- Truth table enumeration (always exponential in n)
- Improved backtracking, e.g., Davis–Putnam–Logemann–Loveland
- Heuristic search in model space (incomplete)
e.g., min-conflicts-like hill-climbing algorithms

Truth table enumeration

The proof system $\mathcal{T}\mathcal{T}$ is specified as follows:

$\{\alpha_1, \dots, \alpha_m\} \vdash_{\mathcal{T}\mathcal{T}} \varphi$ iff all the values in the truth table of $(\alpha_1 \wedge \dots \wedge \alpha_m) \Rightarrow \varphi$ are true

Inference by Truth Tables

- The truth-tables-based inference system is sound:

$\alpha_1, \dots, \alpha_m \vdash_{\mathcal{T}\mathcal{T}} \varphi$ implies truth table of $(\alpha_1 \wedge \dots \wedge \alpha_m) \Rightarrow \varphi$ all true
implies $(\alpha_1 \wedge \dots \wedge \alpha_m) \Rightarrow \varphi$ is valid
implies $\models (\alpha_1 \wedge \dots \wedge \alpha_m) \Rightarrow \varphi$
implies $(\alpha_1 \wedge \dots \wedge \alpha_m) \models \varphi$
implies $\alpha_1, \dots, \alpha_m \models \varphi$

- It is also complete (exercise: prove it)
- Its time complexity is $O(2^n)$ where n is the number of propositional variables in $\alpha_1, \dots, \alpha_m, \varphi$
- We cannot hope to do better in general because the dual problem: determining the satisfiability of a sentence, is NP-complete
- However, really hard cases of propositional inference are somewhat rare in practice

Rule-Based Inference in PL

An inference system in Propositional Logic can also be specified as a set \mathcal{R} of inference (or derivation) rules

- Each rule is just a *pattern* premises/conclusion
- A rule **applies** to Γ and **derives** φ if
 - some of the sentences in Γ match with the premises of the rule and
 - φ matches with the conclusion
- A rule is **sound** if the set of its premises entails its conclusion

Some Inference Rules

- And-Introduction

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

- And-Elimination

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

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

- Or-Introduction

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

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

Some Inference Rules (cont'd)

- Implication-Elimination (aka Modus Ponens)

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

- Unit Resolution

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

- Resolution

$$\frac{\alpha \vee \beta \quad \neg\beta \vee \gamma}{\alpha \vee \gamma} \quad \text{or, equivalently,}$$

$$\frac{\neg\alpha \Rightarrow \beta, \quad \beta \Rightarrow \gamma}{\neg\alpha \Rightarrow \gamma}$$

Some Inference Rules (cont'd)

- Double-Negation-Elimination

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

- False-Introduction

$$\frac{\alpha \wedge \neg\alpha}{\mathbf{False}}$$

- False-Elimination

$$\frac{\mathbf{False}}{\beta}$$

Inference by Proof

We say there is a **proof** of φ from Γ in an inference system \mathcal{R} if we can derive φ by applying the rules of \mathcal{R} repeatedly to Γ and its derived sentences

Example: a proof of P from $\{(P \vee H) \wedge \neg H\}$

1. $(P \vee H) \wedge \neg H$ *by assumption*
2. $P \vee H$ *by \wedge -elimination applied to (1)*
3. $\neg H$ *by \wedge -elimination applied to (1)*
4. P *by unit resolution applied to (2),(3)*

We can represent a proof more visually as a **proof tree**:

Example:

$$\frac{\frac{(P \vee H) \wedge \neg H}{P \vee H} \quad \frac{(P \vee H) \wedge \neg H}{\neg H}}{P}$$

Rule-Based Inference in Propositional Logic

More precisely, there is a proof of φ from Γ in \mathcal{R} if

1. $\varphi \in \Gamma$ or,
2. there is a rule in \mathcal{R} that applies to Γ and produces φ or,
3. there is a proof of each $\varphi_1, \dots, \varphi_m$ from Γ in \mathcal{R} and a rule that applies to $\{\varphi_1, \dots, \varphi_m\}$ and produces φ

Then, the inference system \mathcal{R} is specified as follows:

$\Gamma \vdash_{\mathcal{R}} \varphi$ iff there is a proof of φ from Γ in \mathcal{R}

An Inference System \mathcal{R}

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

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

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

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

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

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

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

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

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

$$\frac{\alpha \wedge \neg \alpha}{\mathbf{False}}$$

$$\frac{\mathbf{False}}{\beta}$$

Soundness of \mathcal{R}

The given system \mathcal{R} is sound because all of its rules are

Example: the Resolution rule
$$\frac{\alpha \vee \beta, \quad \neg\beta \vee \gamma}{\alpha \vee \gamma}$$

	α	β	γ	$\neg\beta$	$\alpha \vee \beta$	$\neg\beta \vee \gamma$	$\alpha \vee \gamma$
1.	<i>false</i>	<i>false</i>	<i>false</i>	<i>true</i>	<i>false</i>	<i>true</i>	<i>false</i>
2.	<i>false</i>	<i>false</i>	<i>true</i>	<i>true</i>	<i>false</i>	<i>true</i>	<i>true</i>
3.	<i>false</i>	<i>true</i>	<i>false</i>	<i>false</i>	<i>true</i>	<i>false</i>	<i>false</i>
4.	<i>false</i>	<i>true</i>	<i>true</i>	<i>false</i>	<u><i>true</i></u>	<u><i>true</i></u>	<i>true</i>
5.	<i>true</i>	<i>false</i>	<i>false</i>	<i>true</i>	<u><i>true</i></u>	<u><i>true</i></u>	<i>true</i>
6.	<i>true</i>	<i>false</i>	<i>true</i>	<i>true</i>	<u><i>true</i></u>	<u><i>true</i></u>	<i>true</i>
7.	<i>true</i>	<i>true</i>	<i>false</i>	<i>false</i>	<i>true</i>	<i>false</i>	<i>true</i>
8.	<i>true</i>	<i>true</i>	<i>true</i>	<i>false</i>	<u><i>true</i></u>	<u><i>true</i></u>	<i>true</i>

All the interpretations that satisfy both $\alpha \vee \beta$ and $\neg\beta \vee \gamma$ (4,5,6,8) satisfy $\alpha \vee \gamma$ as well

Soundness of \mathcal{R}

The given system \mathcal{R} is sound because all of its rules are

Exercise: prove that the other inference rules are sound as well

Is \mathcal{R} also complete?

Resolution

Literal: prop. symbol (P) or negated prop. symbol ($\neg P$)

Clause: set of literals $\{l_1, \dots, l_k\}$ (understood as $l_1 \vee \dots \vee l_k$)

Conjunctive Normal Form: set of clauses $\{C_1, \dots, C_n\}$ (understood as $C_1 \wedge \dots \wedge C_n$)

Resolution rule for CNF:

$$\frac{l_1 \vee \dots \vee l_k \vee P \quad \neg P \vee m_1 \vee \dots \vee m_n}{l_1 \vee \dots \vee l_k \vee m_1 \vee \dots \vee m_n}$$

E.g.,

$$\frac{P \vee Q \quad \neg Q}{P} \quad \frac{P \vee Q \quad R \vee \neg Q \vee \neg S}{P \vee R \vee \neg S} \quad \frac{P \vee Q \quad \neg Q \vee P \vee R}{P \vee R}$$

Resolution is sound and **complete** for CNF KBs

Conversion to CNF

Ex.: $A \Leftrightarrow (B \vee C)$

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

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

2. Eliminate \Rightarrow , replacing $\alpha \Rightarrow \beta$ with $\neg\alpha \vee \beta$

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

3. Move \neg inwards using de Morgan's rules and double-negation

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

4. Apply distributivity law (\vee over \wedge) and flatten

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

Resolution Procedure

Proof by contradiction, i.e., show $KB \wedge \neg\alpha$ unsatisfiable

function PL-RESOLUTION(KB, α) **returns** *true* or *false*

clauses \leftarrow the set of clauses in the CNF representation of $KB \wedge \neg\alpha$

new \leftarrow { }

loop do

for each C_i, C_j **in** *clauses* **do**

resolvents \leftarrow PL-RESOLVE(C_i, C_j)

if *resolvents* contains the empty clause **then return** *true*

new \leftarrow *new* \cup *resolvents*

if *new* \subseteq *clauses* **then return** *false*

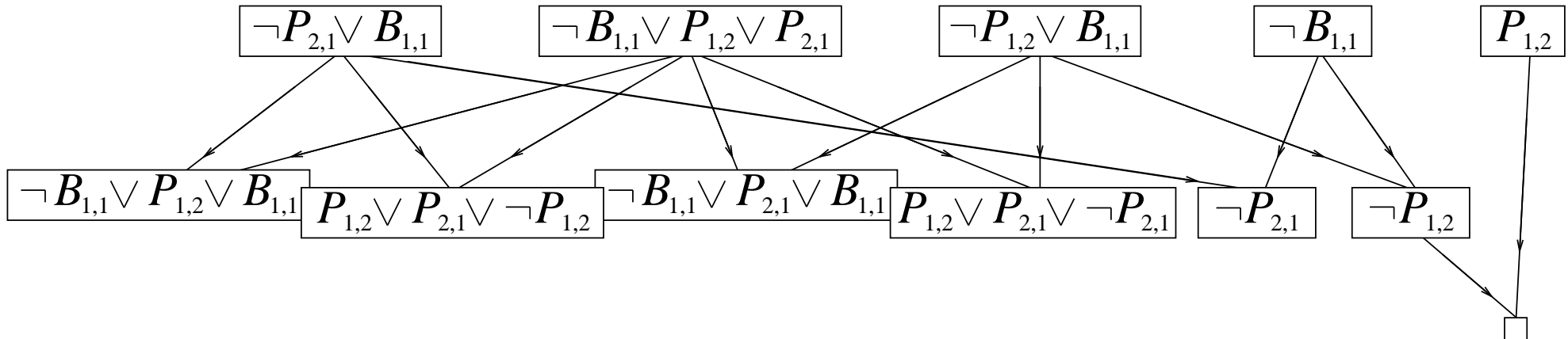
clauses \leftarrow *clauses* \cup *new*

Resolution example

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

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

$$\text{CNF} = \{ \neg P_{1,2} \vee B_{1,1}, \neg B_{1,1} \vee P_{1,2} \vee P_{2,1}, \neg P_{1,2} \vee B_{1,1}, \neg B_{1,1}, P_{1,2} \}$$



Forward and backward chaining

Horn clause: prop. symbol (p) or implication $p_1 \wedge \dots \wedge p_n \Rightarrow p$

Horn Form: set of Horn clauses $\{C_1, \dots, C_n\}$ (understood as $C_1 \wedge \dots \wedge C_n$)

E.g., $\{ C, B \Rightarrow A, C \wedge D \Rightarrow B \}$

Modus Ponens for Horn Form

$$\frac{\alpha_1 \quad \dots \quad \alpha_n \quad \alpha_1 \wedge \dots \wedge \alpha_n \Rightarrow \beta}{\beta}$$

Sound and complete for Horn Form KBs

Can be used with [forward chaining](#) or [backward chaining](#)

These algorithms are very natural and run in **linear** time

Forward chaining

Idea:

Fire any rule whose premises are satisfied in the KB, add its conclusion to the KB, until query is found

Ex.:

$$P \implies Q$$

$$L \wedge M \implies P$$

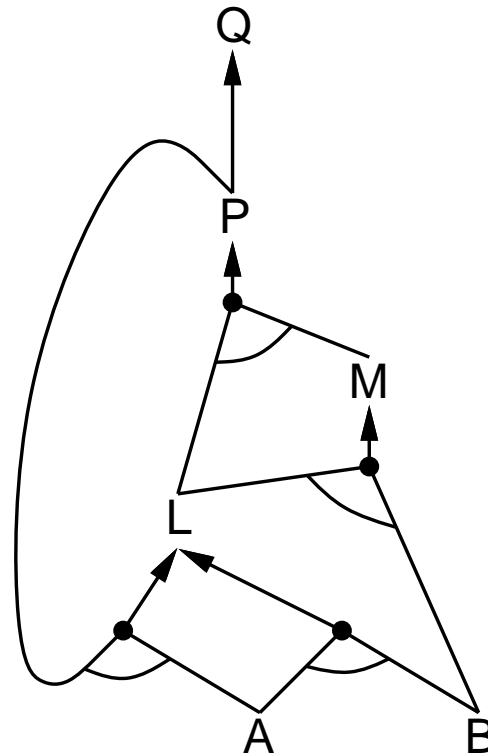
$$B \wedge L \implies M$$

$$A \wedge P \implies L$$

$$A \wedge B \implies L$$

A

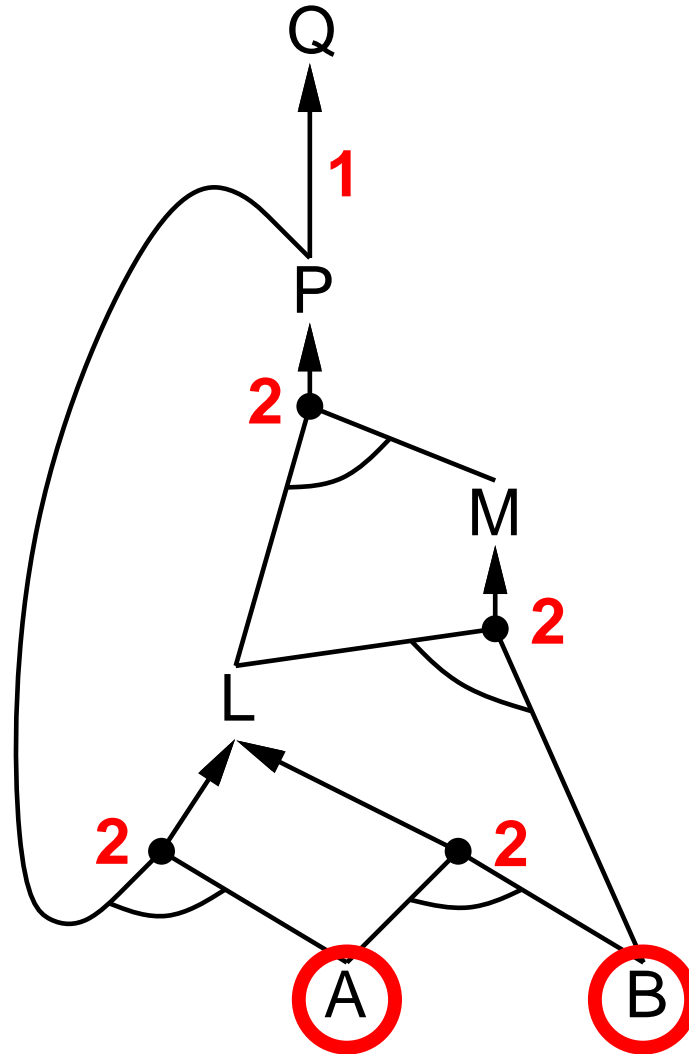
B



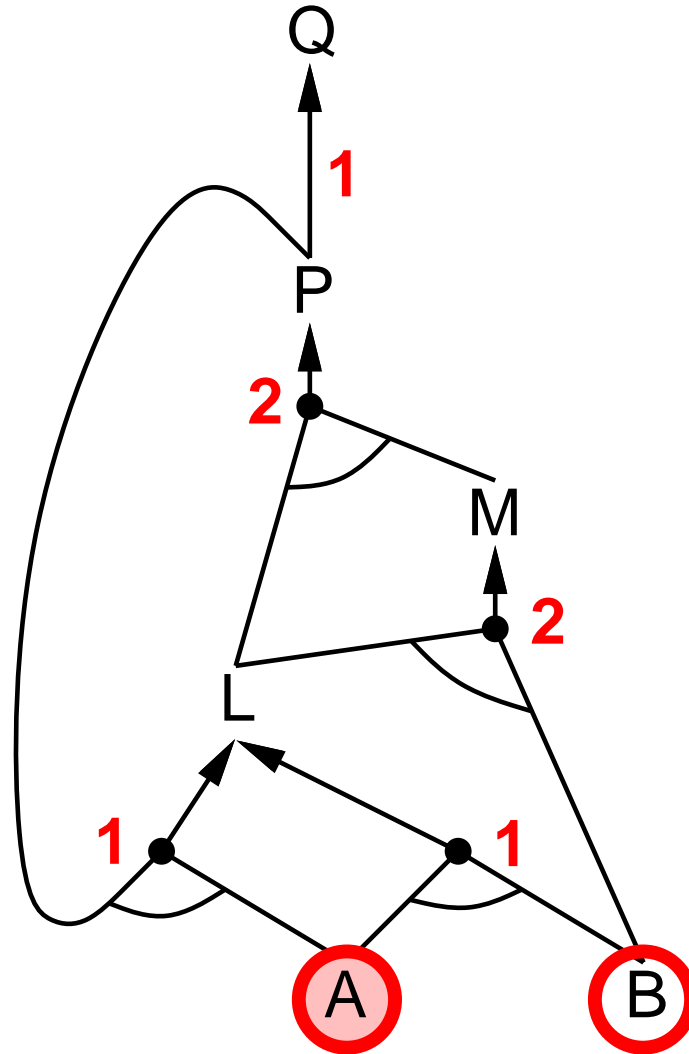
Forward chaining algorithm

```
function PL-FC-ENTAILS?(KB, q) returns true or false  
  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  
     $p \leftarrow \text{POP}(\textit{agenda})$   
    unless inferred[p] do  
       $\textit{inferred}[p] \leftarrow \textit{true}$   
      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
```

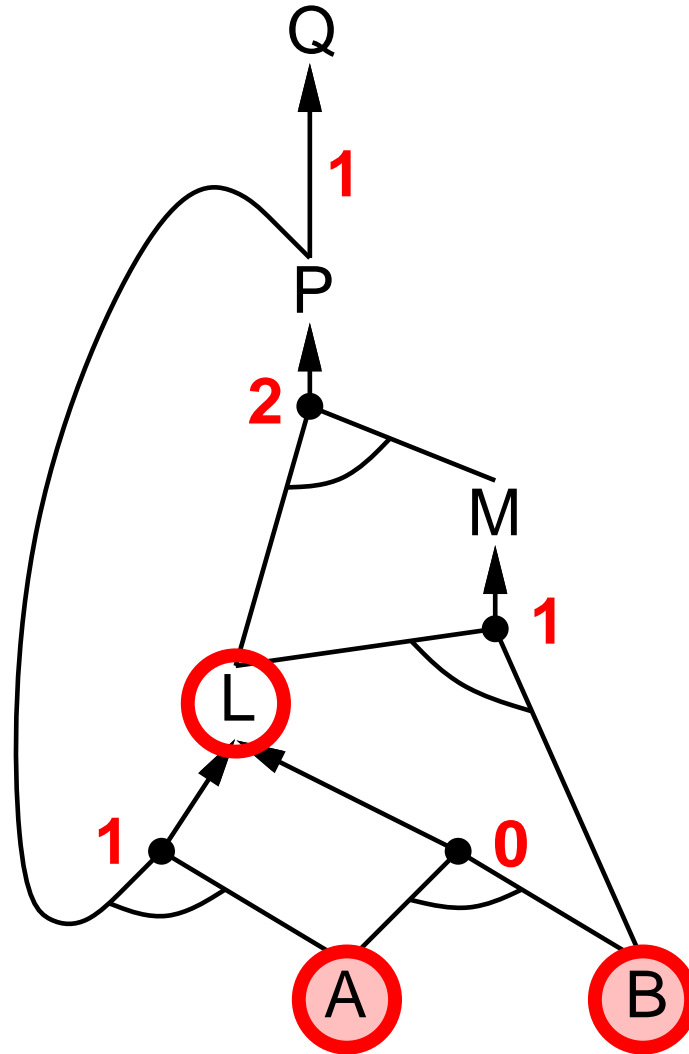
Forward chaining example



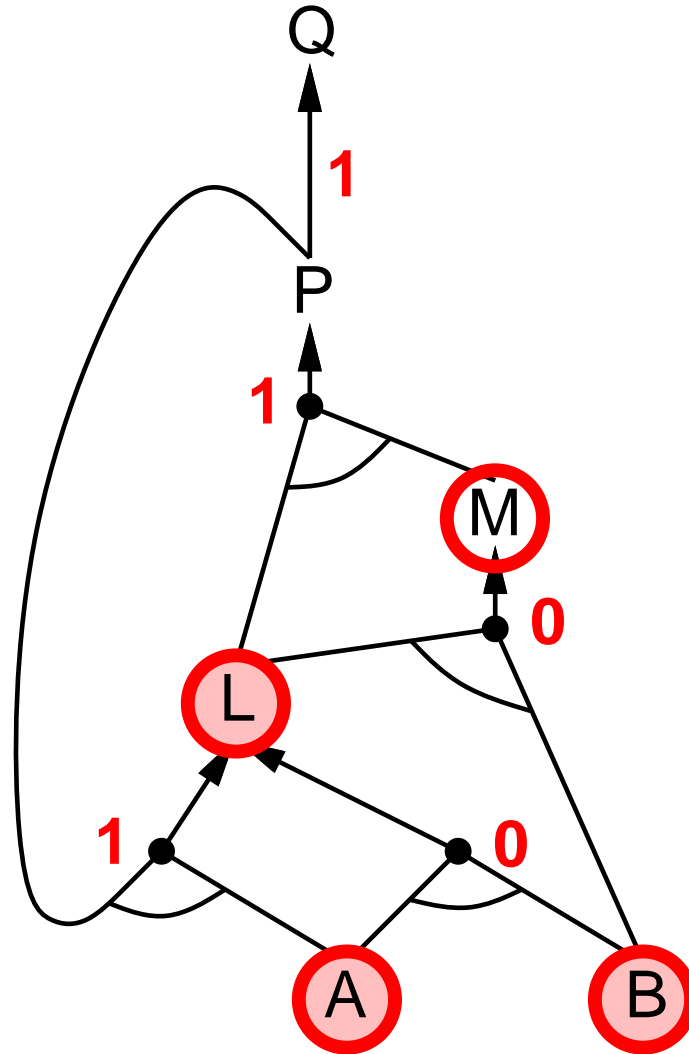
Forward chaining example



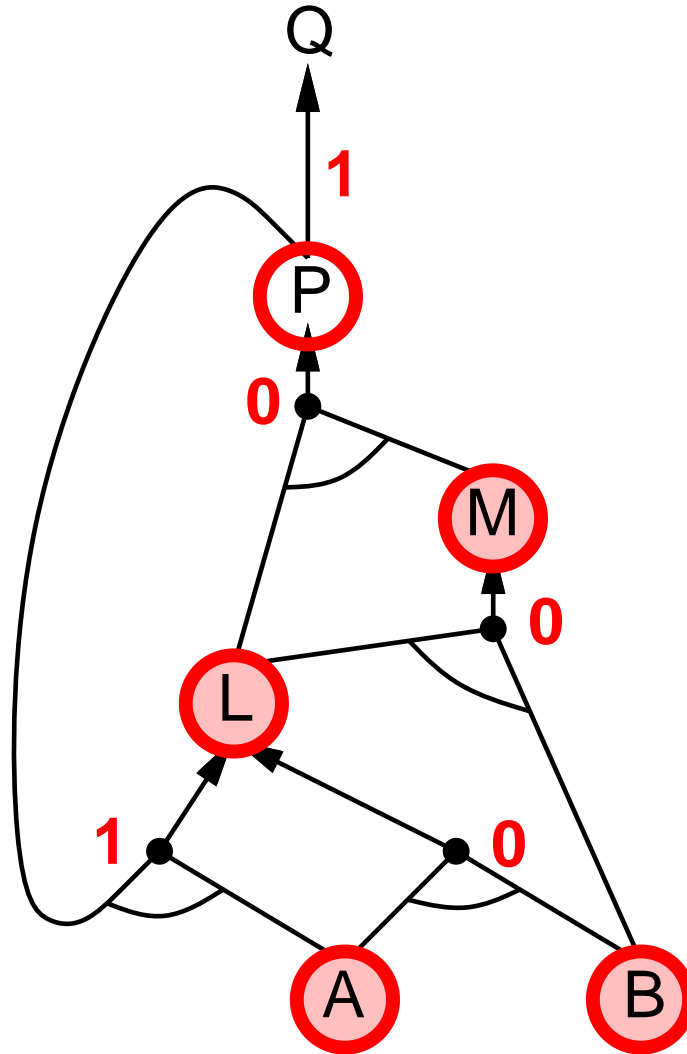
Forward chaining example



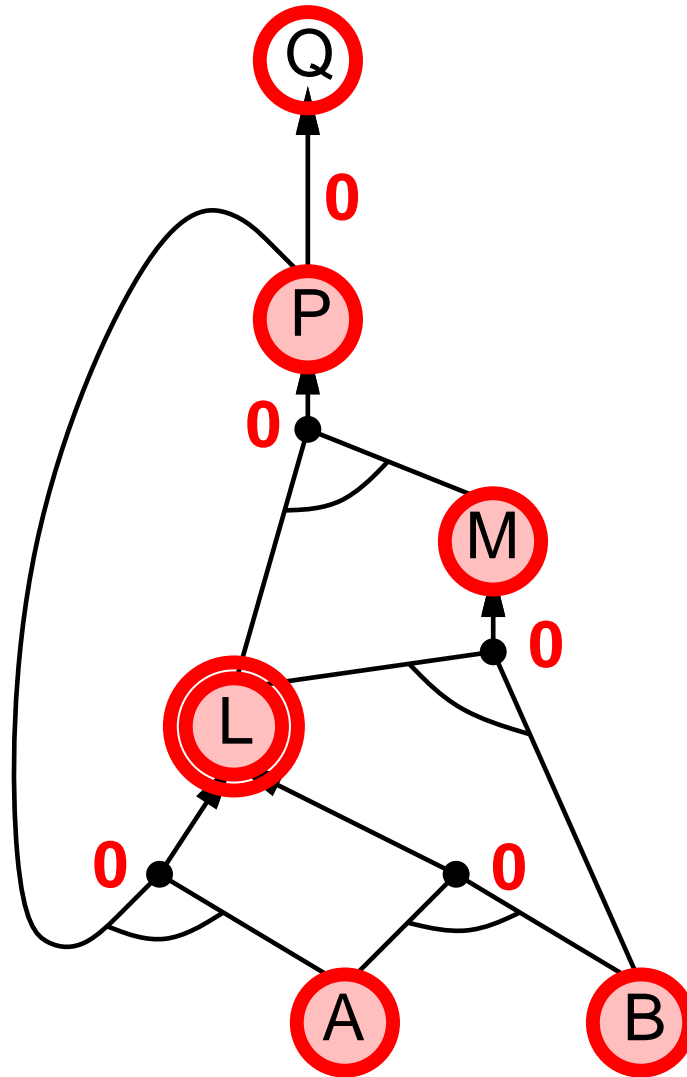
Forward chaining example



Forward chaining example



Forward chaining example



Proof of completeness

FC derives every atomic sentence that is entailed by KB

1. FC reaches a **fixed point** where no new atomic sentences (prop. symbols) are inferred
2. Consider the final state as a model m , assigning true to the inferred symbols and false to the other symbols

3. **Claim:** Every clause in the original KB is satisfied by m

Proof: Suppose a clause $a_1 \wedge \dots \wedge a_k \Rightarrow b$ is falsified by m

Then $a_1 \wedge \dots \wedge a_k$ is satisfied by m while b is not

But then b was not inferred, contradicting the assumption that the algorithm had reached a fixed point!

4. Hence m is a model of KB

5. If $KB \models q$, q is true in **every** model of KB, including m

Backward chaining

Idea: work backwards from the query q

to infer q by BC,

check if q is known already, or

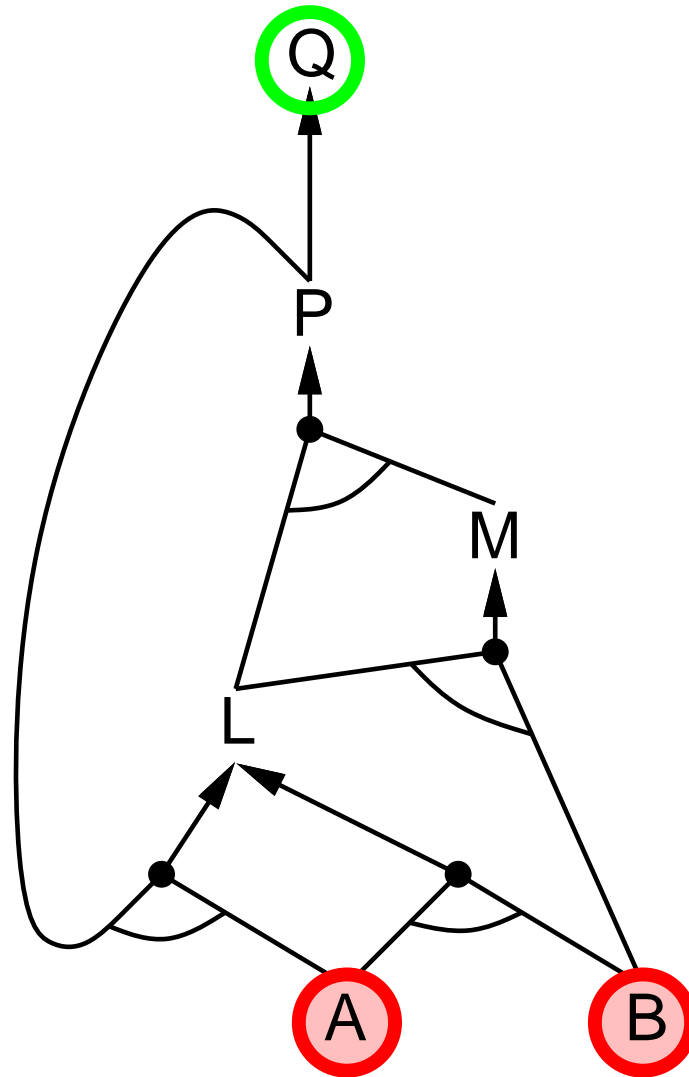
infer by BC all premises of some rule concluding q

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

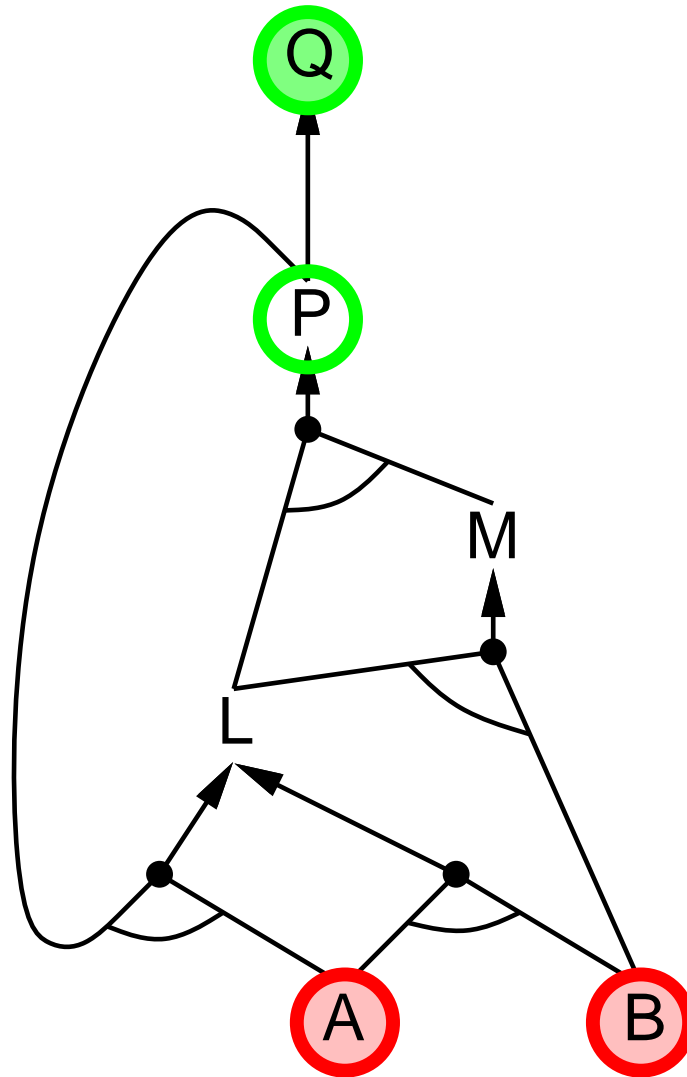
Avoid repeated work: check if new subgoal

1. has already been inferred, or
2. has already failed

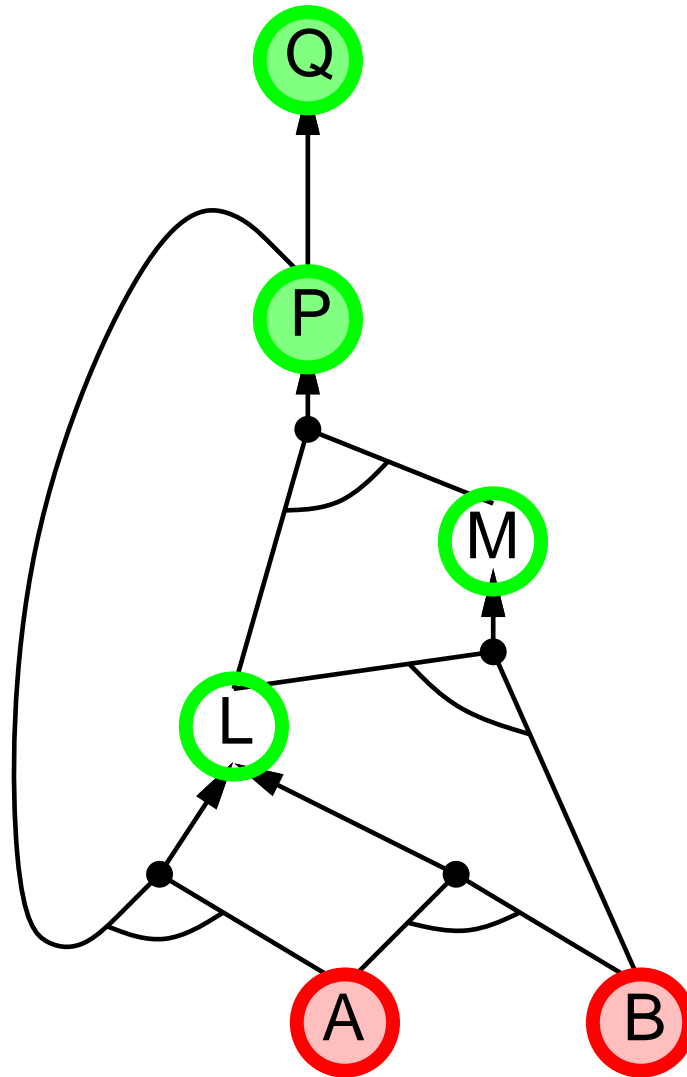
Backward chaining example



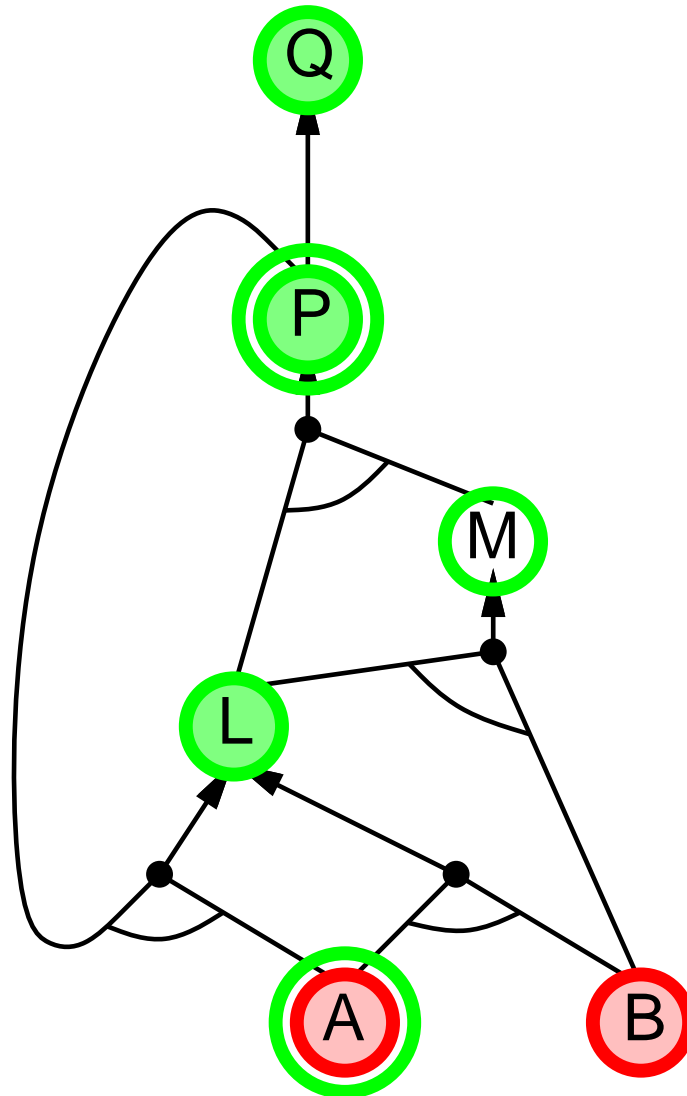
Backward chaining example



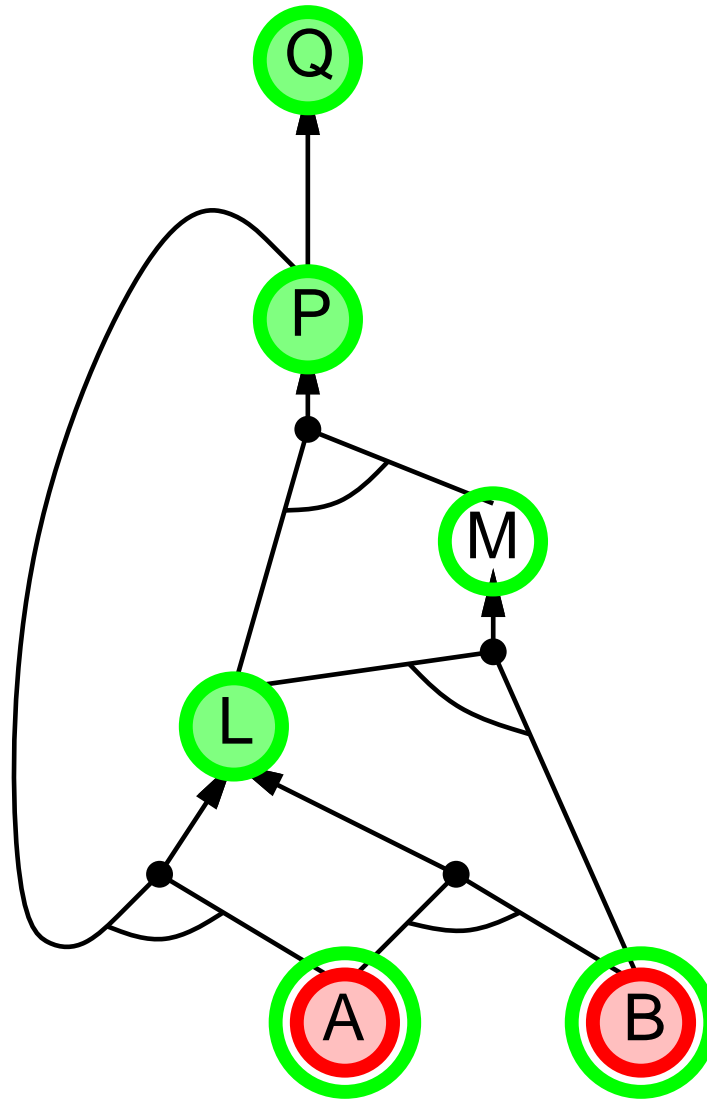
Backward chaining example



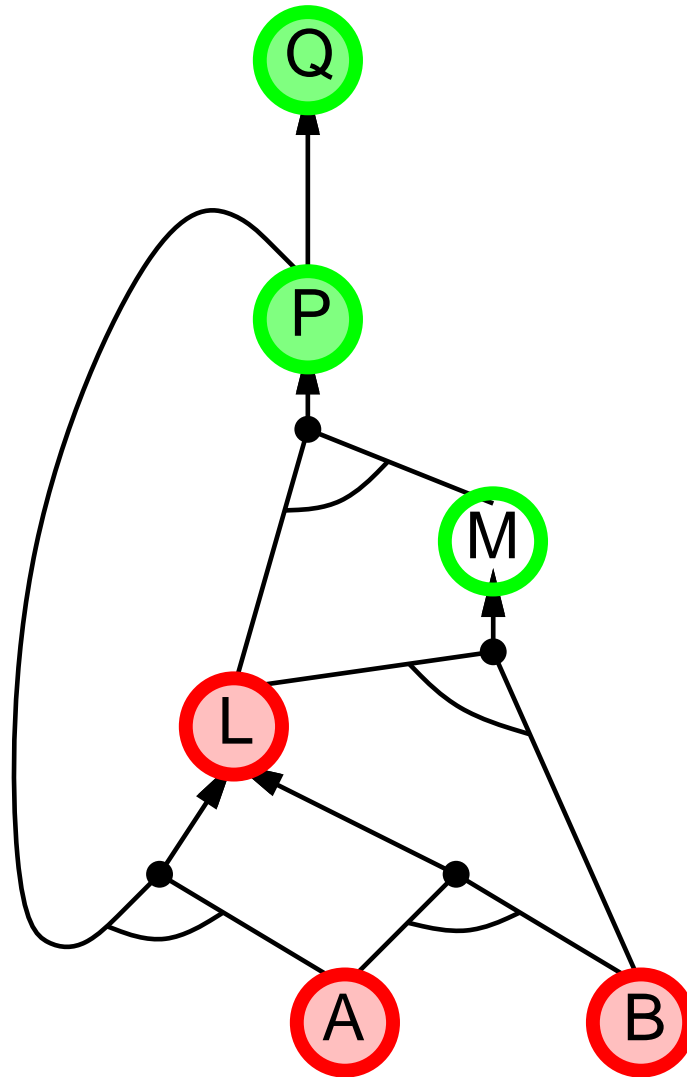
Backward chaining example



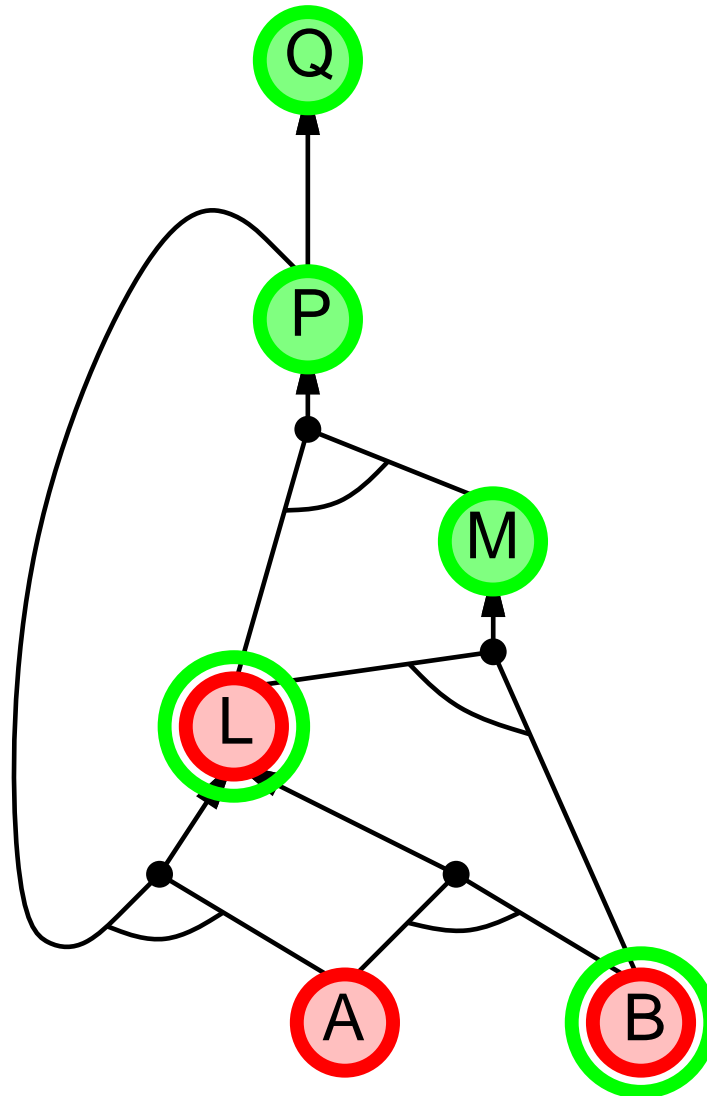
Backward chaining example



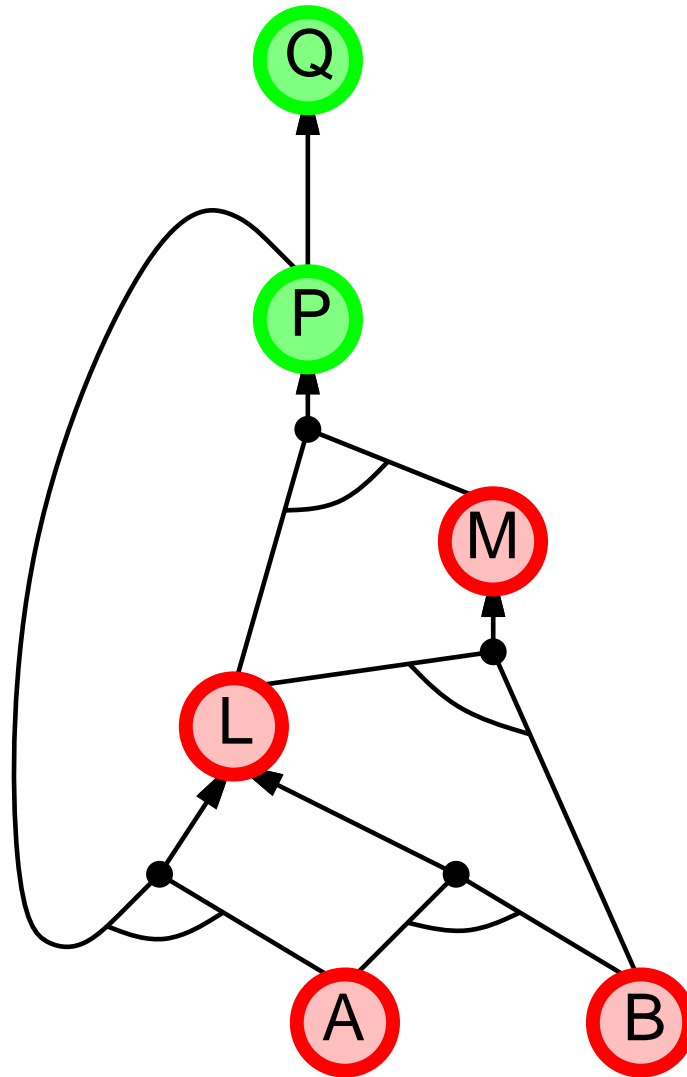
Backward chaining example



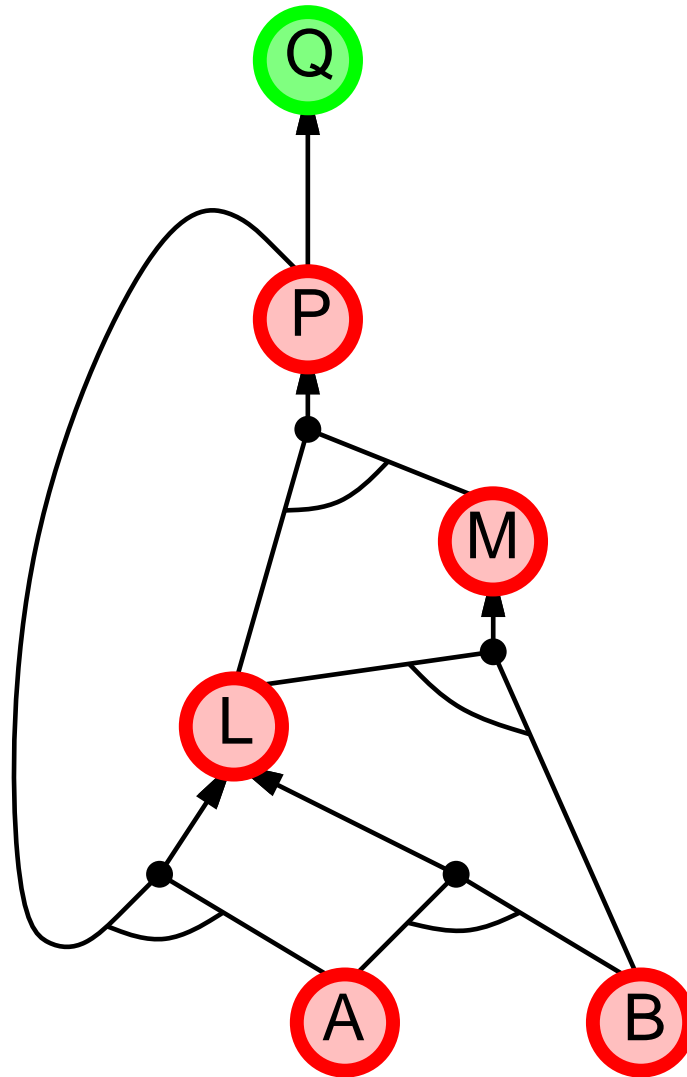
Backward chaining example



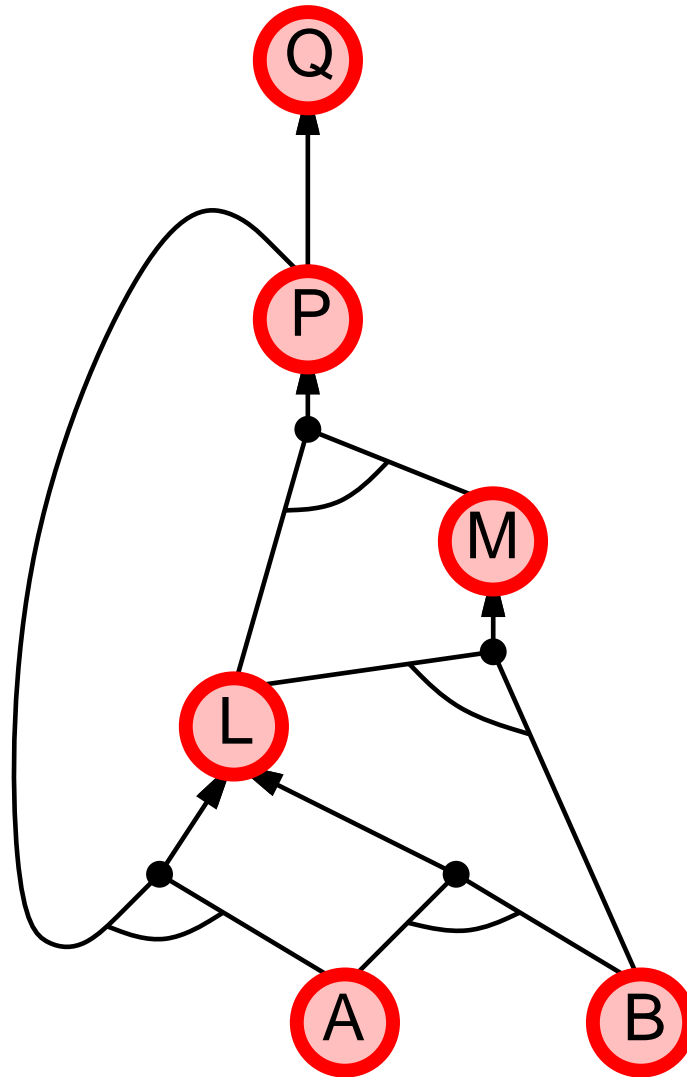
Backward chaining example



Backward chaining example



Backward chaining example



Forward vs. Backward Chaining

FC is **data-driven**, cf. 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

Model Checking methods

The most effective procedures for propositional satisfiability are based on CSP techniques

Variable domain: $\{true, false\}$

Constraints: sets of clauses

Heuristic Improvements:

- unit propagation
- variable and value ordering
- intelligent backtracking
- clause learning
- random restarts
- clever indexing
- subproblem decomposition

DPLL Procedure

function DPLL-SATISFIABLE?(*s*) **returns** *true* or *false*

inputs: *s*, a sentence in propositional logic

clauses \leftarrow the set of clauses in the CNF representation of *s*

symbols \leftarrow a list of the proposition symbols in *s*

return DPLL(*clauses*, *symbols*, [])

DPLL Procedure (cont.)

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

if every clause in *clauses* is satisfied by *model* **then return** *true*

if some clause in *clauses* is falsified by *model* **then return** *false*

P, *value* \leftarrow FIND-PURE-SYMBOL(*symbols*, *clauses*, *model*)

if *P* is non-null **then**

return DPLL(*clauses*, *symbols* - *P*, (*P* \mapsto *value*) :: *model*)

P, *value* \leftarrow FIND-UNIT-CLAUSE(*clauses*, *model*)

if *P* is non-null **then**

return DPLL(*clauses*, *symbols* - *P*, (*P* \mapsto *value*) :: *model*)

P \leftarrow FIRST(*symbols*)

rest \leftarrow REST(*symbols*)

return DPLL(*clauses*, *rest*, (*P* \mapsto *true*) :: *model*) **or**

 DPLL(*clauses*, *rest*, (*P* \mapsto *false*) :: *model*)

DPLL Exercise

- (1) $\neg p_1 \vee p_2$ (2) $\neg p_3 \vee p_4$ (3) $\neg p_6 \vee \neg p_5 \vee \neg p_2$
(4) $\neg p_5 \vee p_6$ (5) $p_5 \vee p_7$ (6) $\neg p_1 \vee p_5 \vee \neg p_7$