Propositional Logic: Methods of Proof (Part II)

This lecture topic:

Propositional Logic (two lectures)

Chapter 7.1-7.4 (previous lecture, Part I)

Chapter 7.5 (this lecture, Part II)

Next lecture topic:

First-order logic (two lectures)

Chapter 8

(Please read lecture topic material before and after each lecture on that topic)

Outline

- Basic definitions
  - Inference, derive, sound, complete
- Application of inference rules
  - Resolution
  - Horn clauses
  - ~~Forward & Backward chaining~~
- Model Checking
  - ~~Complete backtracking search algorithms~~
    - E.g., DPLL algorithm
  - ~~Incomplete local search algorithms~~
    - E.g., WalkSAT algorithm

You will be expected to know

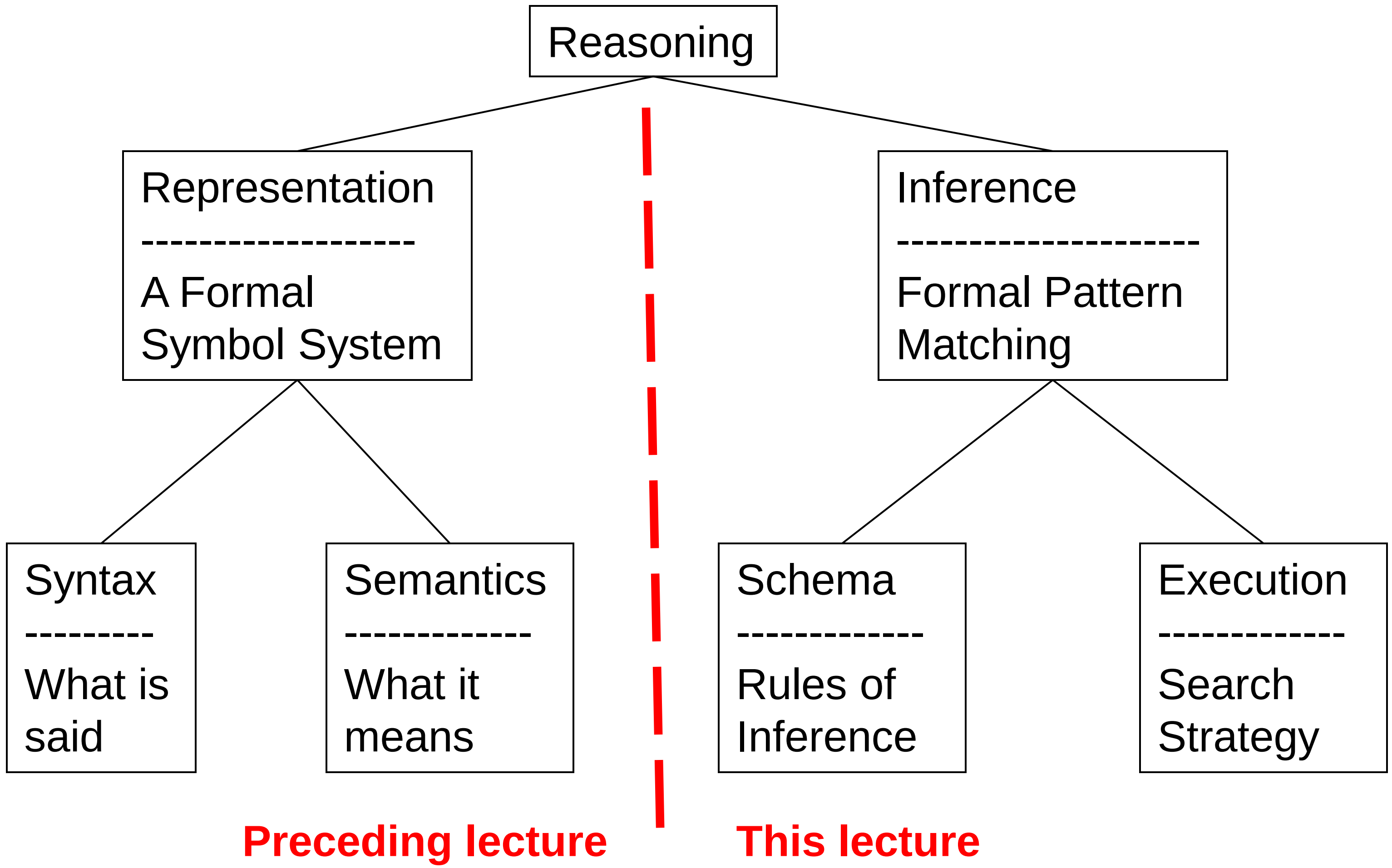
- Basic definitions
- Conjunctive Normal Form (CNF)
  - Convert a Boolean formula to CNF
- Do a short resolution proof
- ~~Do a short forward-chaining proof~~
- ~~Do a short backward-chaining proof~~
- ~~Model checking with backtracking search~~
- ~~Model checking with local search~~

Inference in Formal Symbol Systems: Ontology, Representation, Inference

- **Formal Symbol Systems**
  - **Symbols** correspond to **things/ideas** in the world
  - **Pattern matching & rewrite** corresponds to **inference**

- **Ontology:** What exists in the world?
  - What must be represented?
- **Representation:** Syntax vs. Semantics
  - What’s Said vs. What’s Meant
- **Inference:** Schema vs. Mechanism
  - Proof Steps vs. Search Strategy

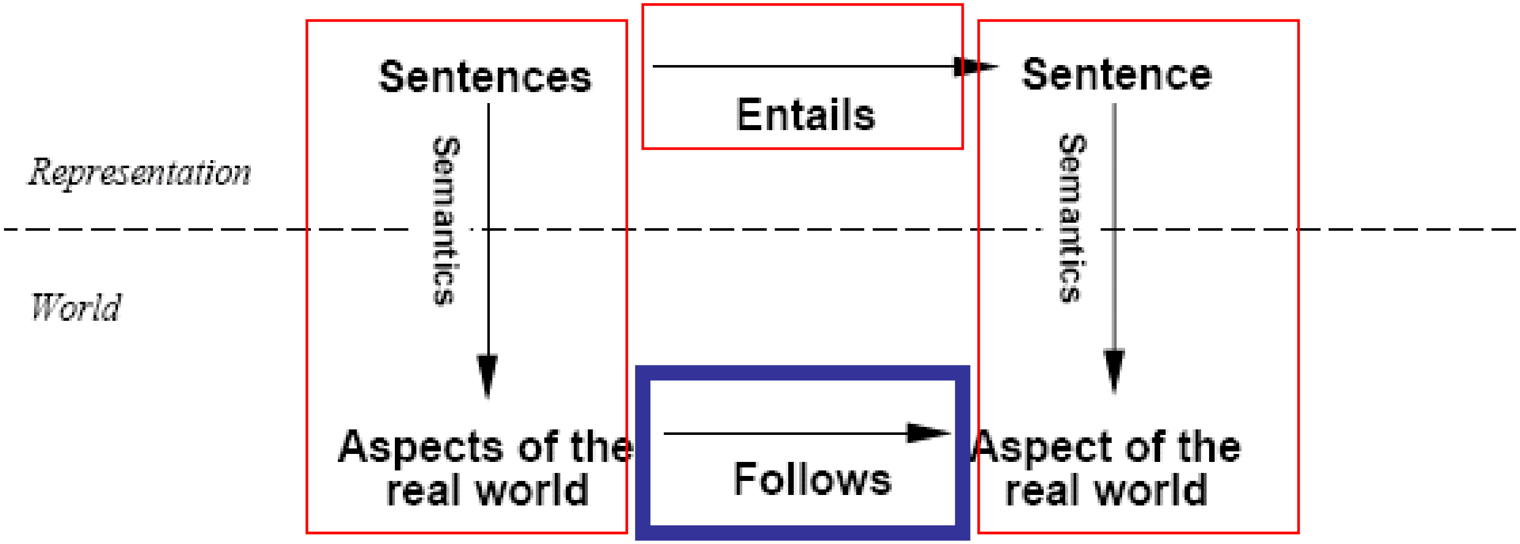
Ontology:
What kind of things exist in the world?
What do we need to describe and reason about?



Review

- Definitions:
  - Syntax, Semantics, Sentences, Propositions, Entails, Follows, Derives, Inference, Sound, Complete, Model, Satisfiable, Valid (or Tautology)
- Syntactic Transformations:
  - E.g., $(A \Rightarrow B) \Leftrightarrow (\neg A \vee B)$
- Semantic Transformations:
  - E.g., $(KB \models \alpha) \equiv (\models (KB \Rightarrow \alpha))$
- Truth Tables
  - Negation, Conjunction, Disjunction, Implication, Equivalence (Biconditional)
  - Inference by Model Enumeration

Review: Schematic perspective



If KB is true in the real world,

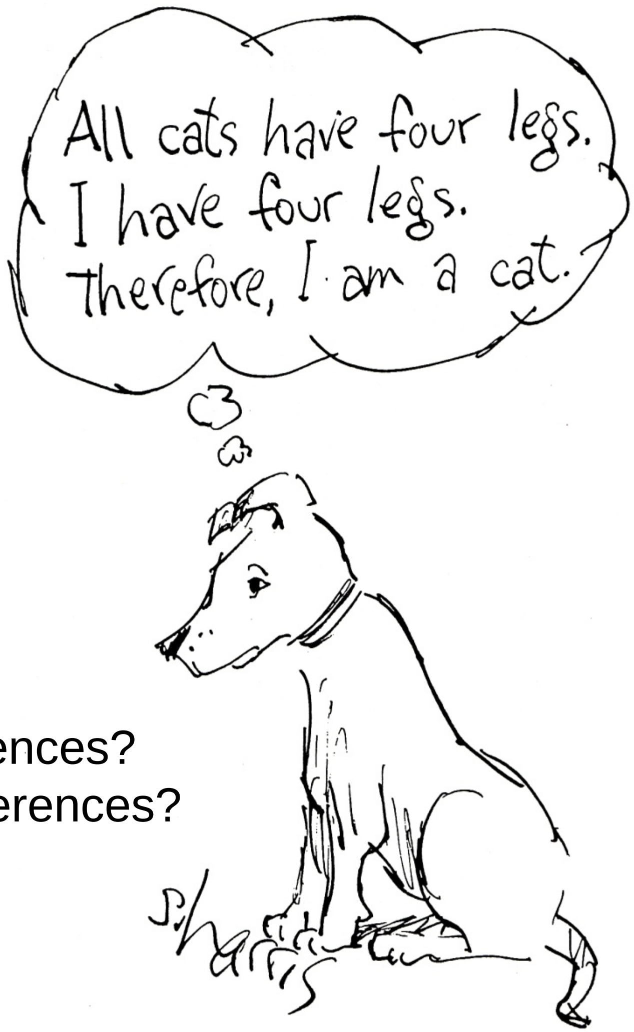
then any sentence α ***entailed*** *by KB*

is also true in the real world.

So --- how do we keep it from "Just making things up." ?

Is this inference correct?

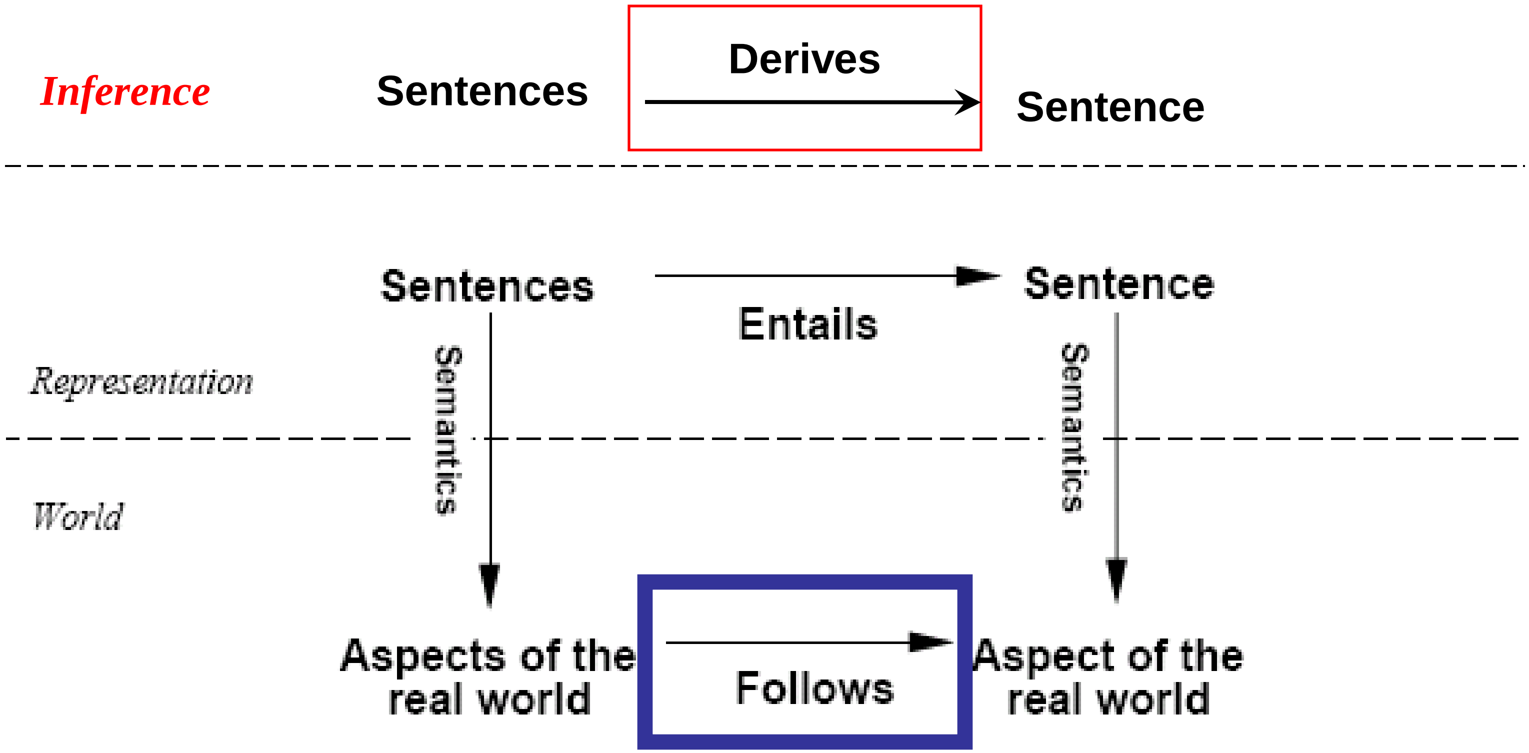
How do you know?
How can you tell?



How can we **make correct** inferences?
How can we **avoid incorrect** inferences?

"Einstein Simplified: Cartoons on Science" by Sydney Harris, 1992, Rutgers University Press

Schematic perspective



If KB is true in the real world,

then any sentence α ***derived*** *from KB*

by a sound inference procedure

is also true in the real world.

Logical inference

- The notion of entailment can be used for logic inference.
  - Model checking (see wumpus example):
    enumerate all possible models and check whether α is true.

- *Sound* (or *truth preserving*):
  The algorithm **only** derives entailed sentences.
  - Otherwise it just makes things up.
    i is sound iff whenever KB $\vdash_i \alpha$ *it is also true that KB* $\models \alpha$
  - *E.g., model-checking is sound*

- Complete:
  The algorithm can derive **every** entailed sentence.
    i is complete iff whenever KB $\models \alpha$ *it is also true that KB* $\vdash_i \alpha$

Proof methods

- Proof methods divide into (roughly) two kinds:

  Application of inference rules:

  Legitimate (sound) generation of new sentences from old.

  – Resolution
  – Forward & Backward chaining

  Model checking

  Searching through truth assignments.

  - Improved backtracking: Davis--Putnam-Logemann-Loveland (DPLL)
  - Heuristic search in model space: Walksat.

Conjunctive Normal Form

We'd like to prove:

$KB \models \alpha$

equivalent to : $KB \wedge \neg\alpha$ *unsatifiable*

We first rewrite $KB \wedge \neg\alpha$ into **conjunctive normal form (CNF).**

A "conjunction of disjunctions"

literals

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

Clause Clause

- Any KB can be converted into CNF.
- In fact, any KB can be converted into CNF-3 using clauses with at most 3 literals.

Example: 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: $\neg(\alpha \vee \beta) = \neg\alpha \wedge \neg\beta$
   $(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge ((\neg P_{1,2} \wedge \neg P_{2,1}) \vee B_{1,1})$

4. Apply distributive law ($\wedge$ over $\vee$) and flatten:
   $(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge (\neg P_{1,2} \vee B_{1,1}) \wedge (\neg P_{2,1} \vee B_{1,1})$

Example: Conversion to CNF

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

5. KB is the conjunction of all of its sentences (all are true), so write each clause (disjunct) as a sentence in KB:

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

Resolution

- Resolution: inference rule for CNF: sound and complete! *

$(A \vee B \vee C)$

$(\neg A)$

“If A or B or C is true, but not A, then B or C must be true.”

$\_\_\_\_\_\_\_\_\_\_\_\_$

$\therefore (B \vee C)$

$(A \vee B \vee C)$

$(\neg A \vee D \vee E)$

“If A is false then B or C must be true, or if A is true then D or E must be true, hence since A is either true or false, B or C or D or E must be true.”

$\_\_\_\_\_\_\_\_\_\_\_\_$

$\therefore (B \vee C \vee D \vee E)$

$(A \vee B)$

$(\neg A \vee B)$

$\_\_\_\_\_\_\_\_\_$

$\therefore (B \vee B) \equiv B$

Simplification

***** Resolution is “refutation complete” in that it can prove the truth of any entailed sentence by refutation.

Resolution Algorithm

- The resolution algorithm tries to prove: $KB \models \alpha$ *equivalent to*
  $KB \wedge \neg\alpha$ *unsatisfiable*

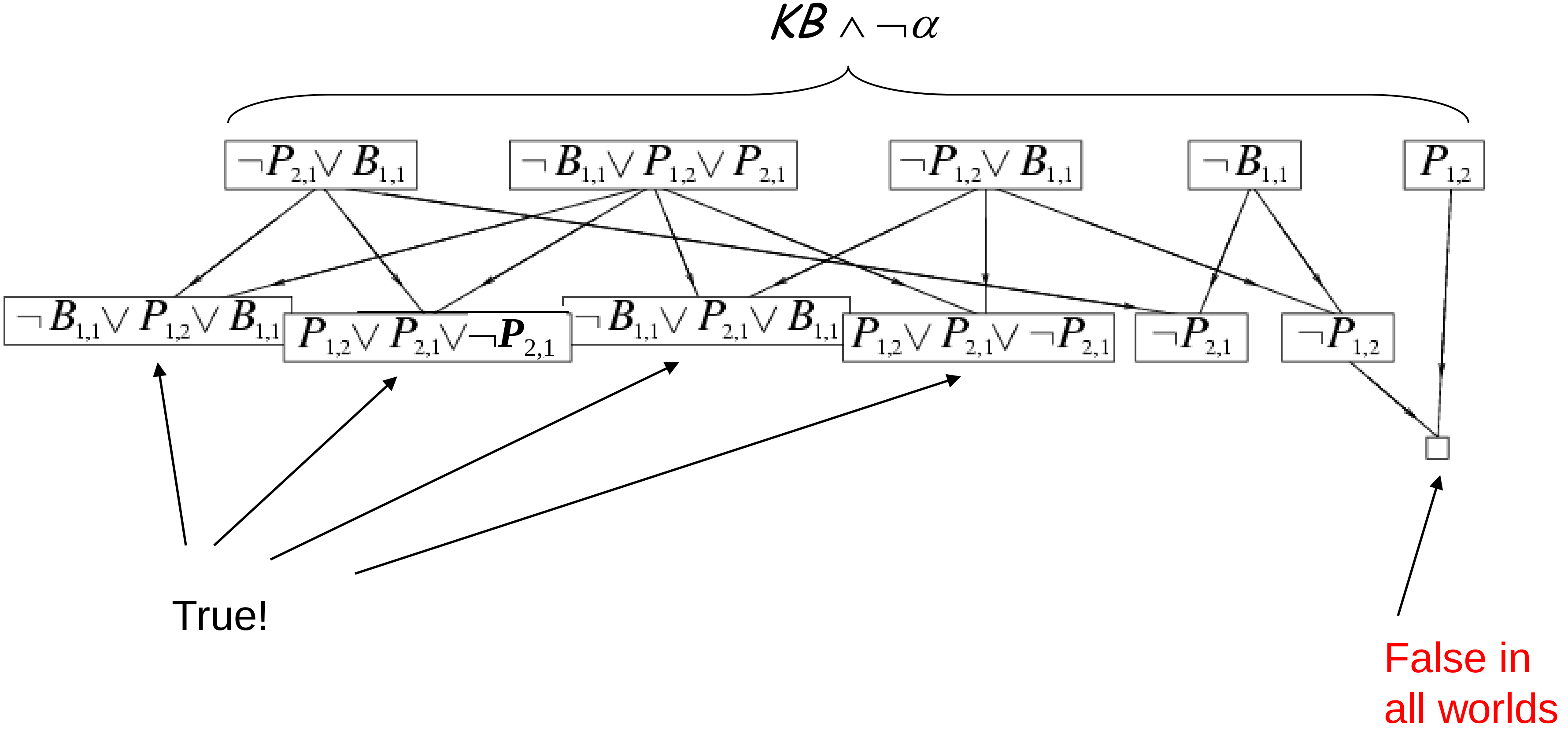
- Generate all new sentences from KB and the (negated) query.
- One of two things can happen:

1. We find $P \wedge \neg P$ which is unsatisfiable. I.e. we can entail the query.

2. We find no contradiction: there is a model that satisfies the sentence $KB \wedge \neg\alpha$ (non-trivial) and hence we cannot entail the query.

Resolution example

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



Try it Yourselves

- 7.9 page 238: (Adapted from Barwise and Etchemendy (1993).) If the unicorn is mythical, then it is immortal, but if it is not mythical, then it is a mortal mammal. If the unicorn is either immortal or a mammal, then it is horned. The unicorn is magical if it is horned.

- *Derive the KB in normal form.*
- *Prove: Horned, Prove: Magical.*

Exposes useful constraints

- **"You can't learn what you can't represent."** --- G. Sussman
- **In logic:** *If the unicorn is mythical, then it is immortal, but if it is not mythical, then it is a mortal mammal. If the unicorn is either immortal or a mammal, then it is horned. The unicorn is magical if it is horned.*

  Prove that the unicorn is both magical and horned.
- **A good representation makes this problem easy:**

$$(\neg Y \vee \neg R) \wedge (Y \vee R) \wedge (Y \vee M) \wedge (R \vee H) \wedge (\neg M \vee H) \wedge (\neg H \vee G)$$

Horn Clauses

- Resolution can be exponential in space and time.

- If we can reduce all clauses to "Horn clauses" resolution is linear in space and time

A clause with at most 1 positive literal.
e.g. $A \vee \neg B \vee \neg C$

- Every Horn clause can be rewritten as an implication with a conjunction of positive literals in the premises and a single positive literal as a conclusion.

e.g. $B \wedge C \Rightarrow A$

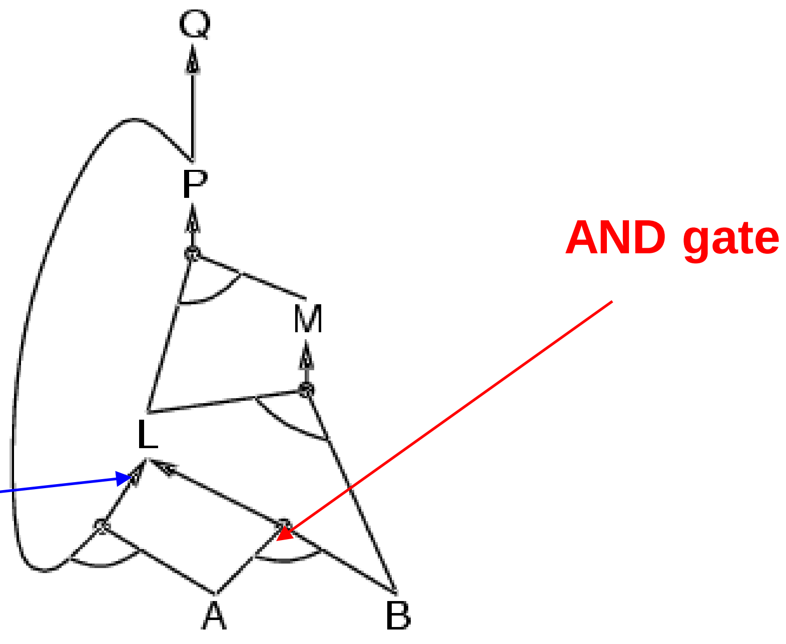
- 1 positive literal: definite clause
- 0 positive literals: integrity constraint:
- e.g. $(\neg A \vee \neg B) \equiv (A \wedge B \Rightarrow False)$
- 0 negative literals: fact
- Forward Chaining and Backward chaining are sound and complete with Horn clauses and run linear in space and time.

Forward chaining (FC)

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

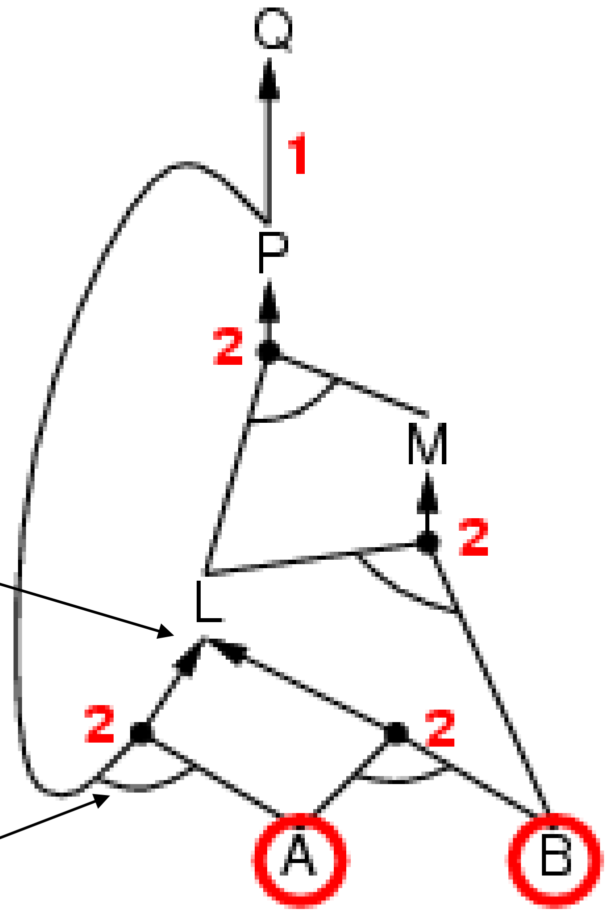
- This proves that $KB \Rightarrow Q$ is true in all possible worlds (i.e. trivial), and hence it proves entailment.

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



- Forward chaining is sound and complete for Horn KB

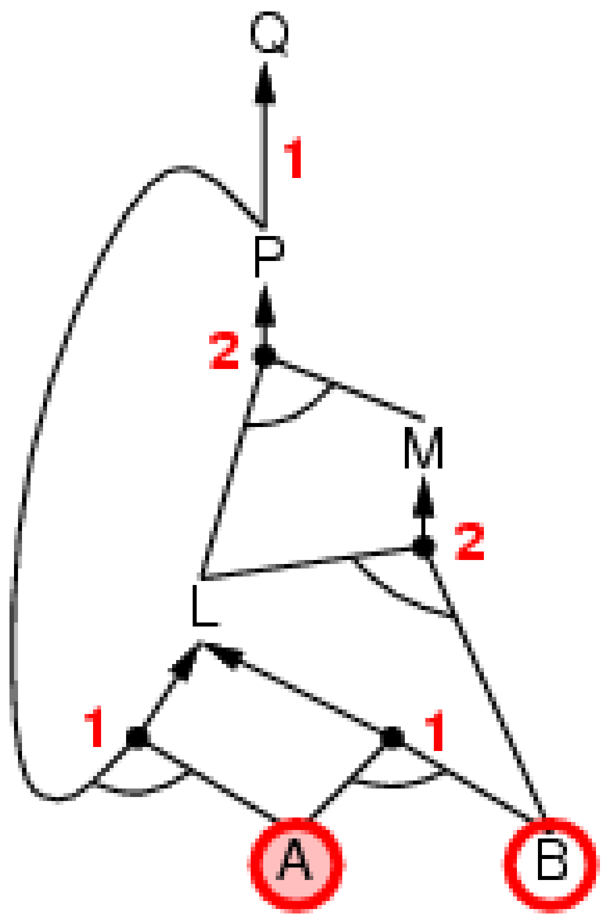
Forward chaining example



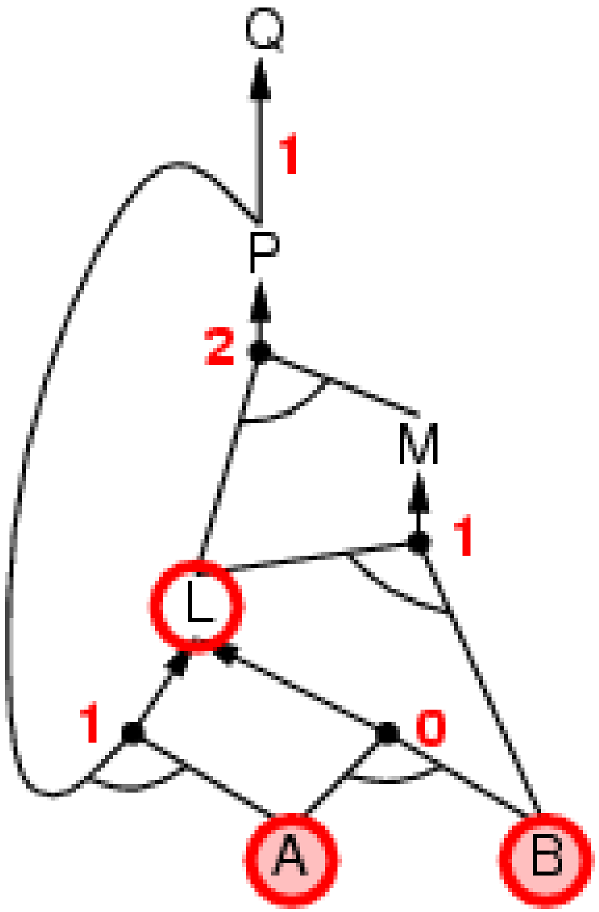
“OR” Gate

“AND” gate

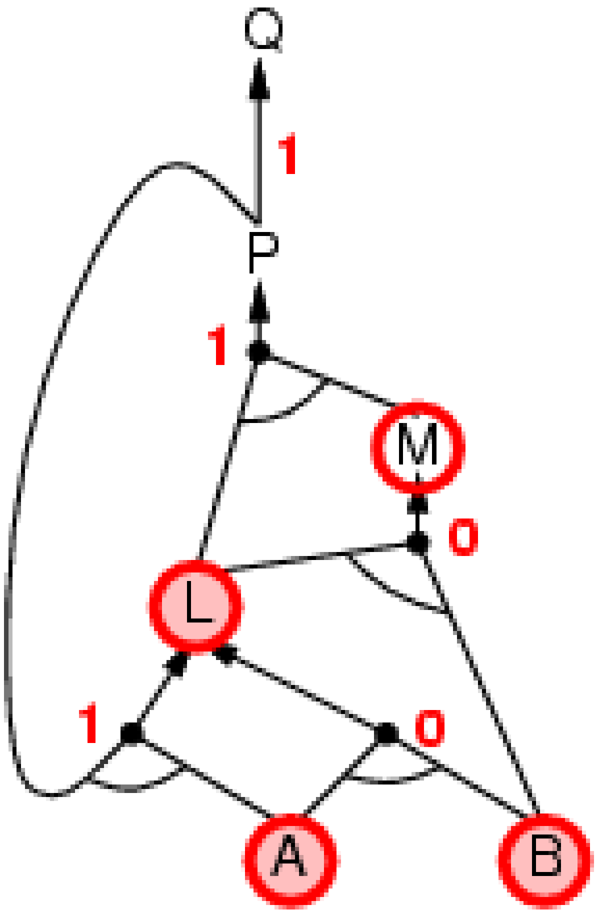
Forward chaining example



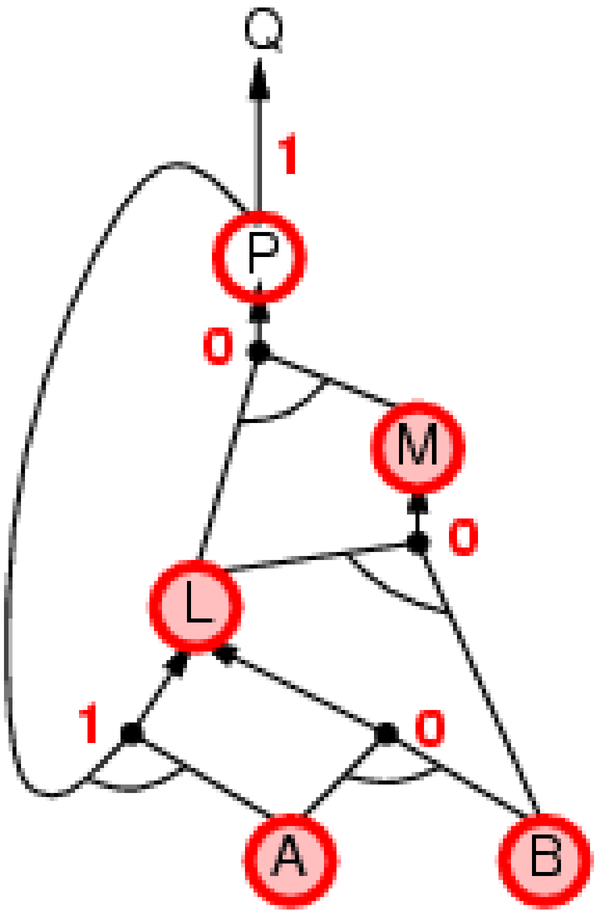
Forward chaining example



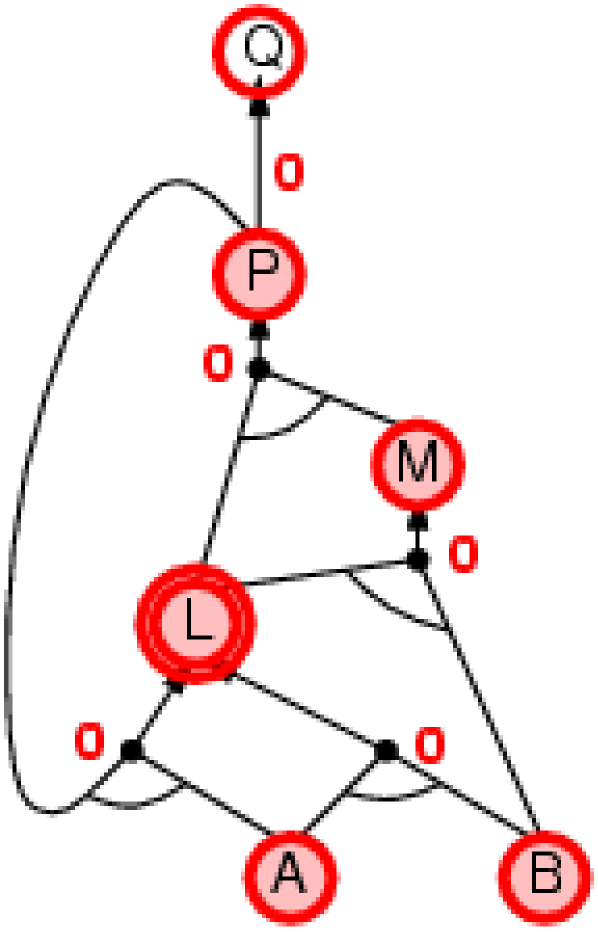
Forward chaining example



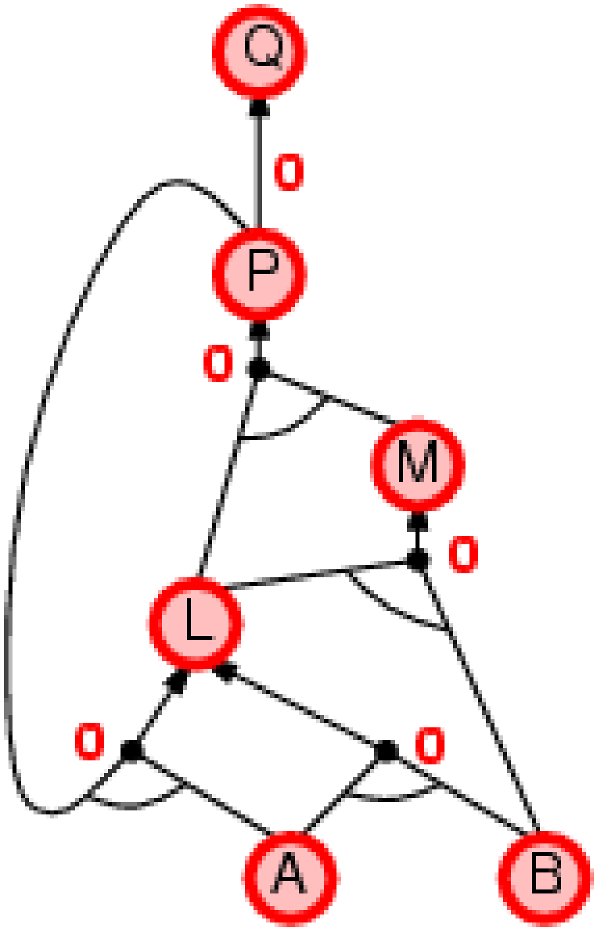
Forward chaining example



Forward chaining example



Forward chaining example



Backward chaining (BC)

Idea: work backwards from the query *q*

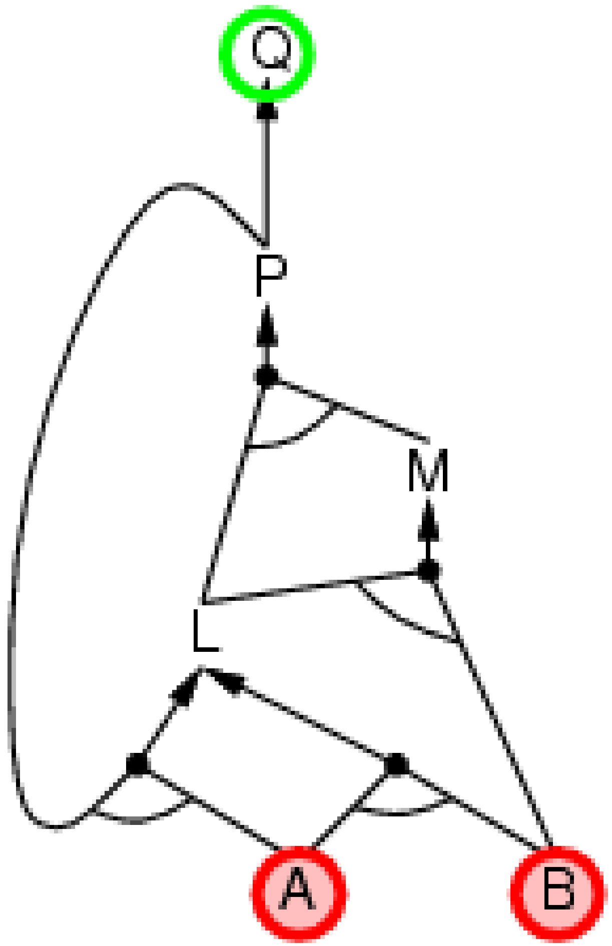
- check if *q* is known already, or
- prove by BC all premises of some rule concluding *q*
- Hence BC maintains a stack of sub-goals that need to be proved to get to q.

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

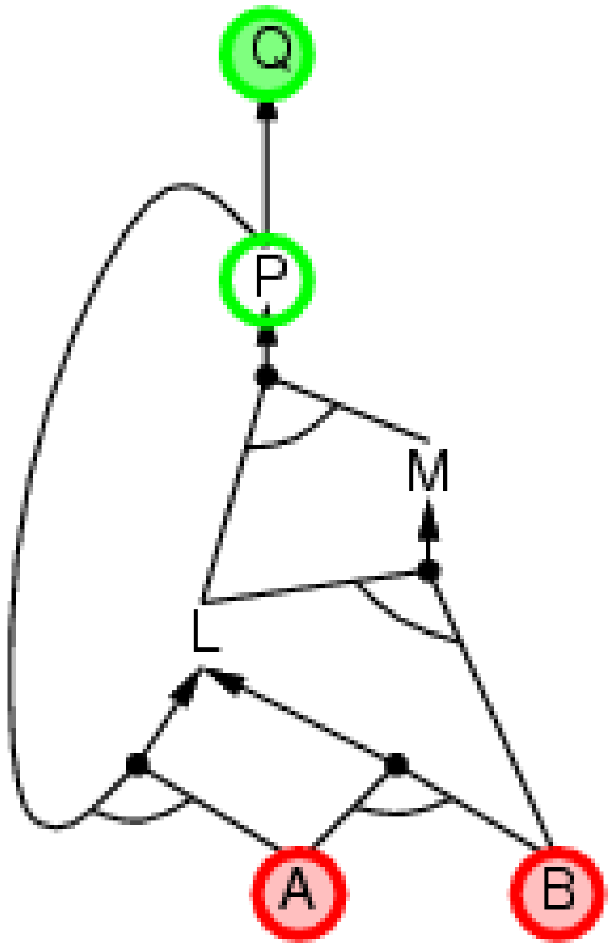
Avoid repeated work: check if new sub-goal

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

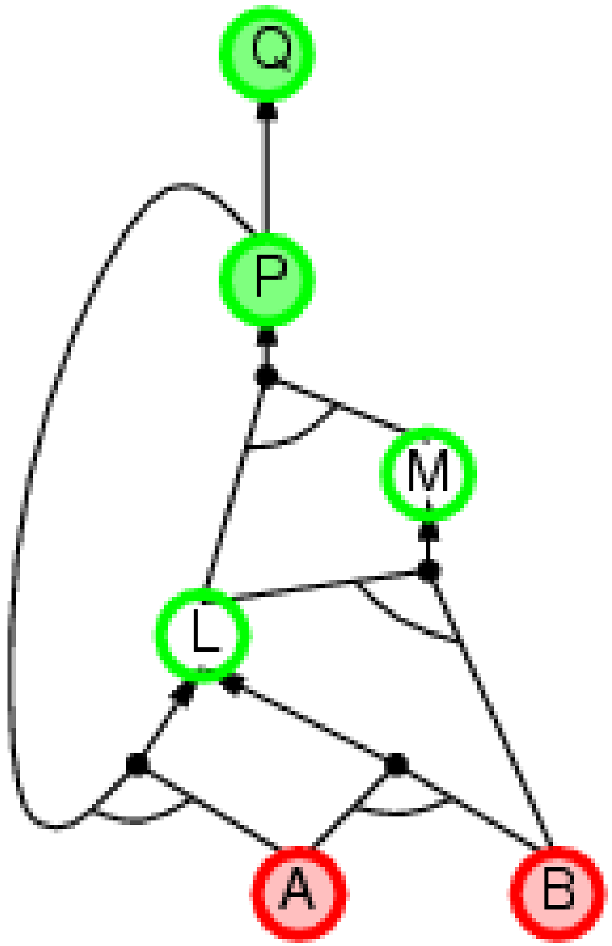
Backward chaining example



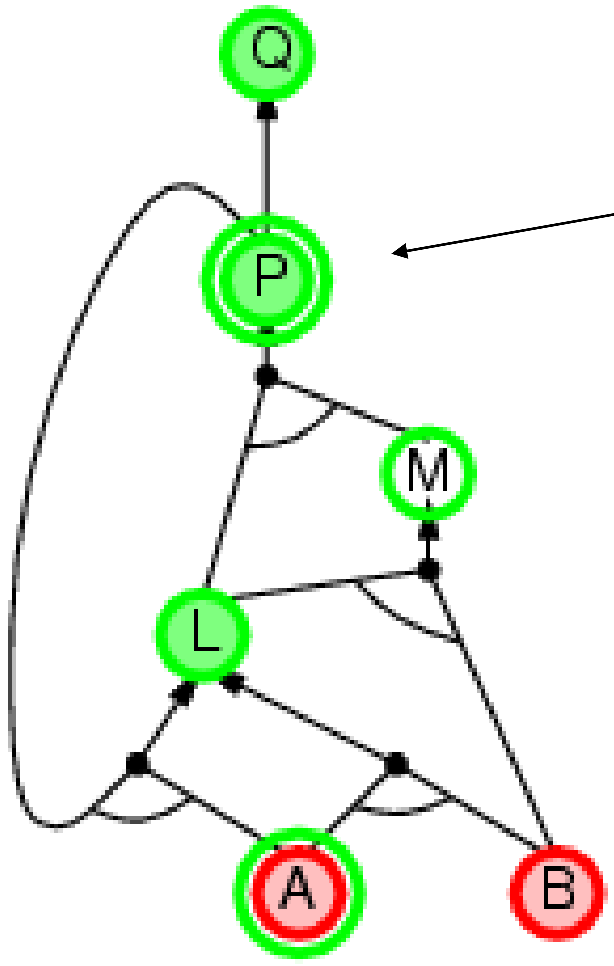
Backward chaining example



Backward chaining example

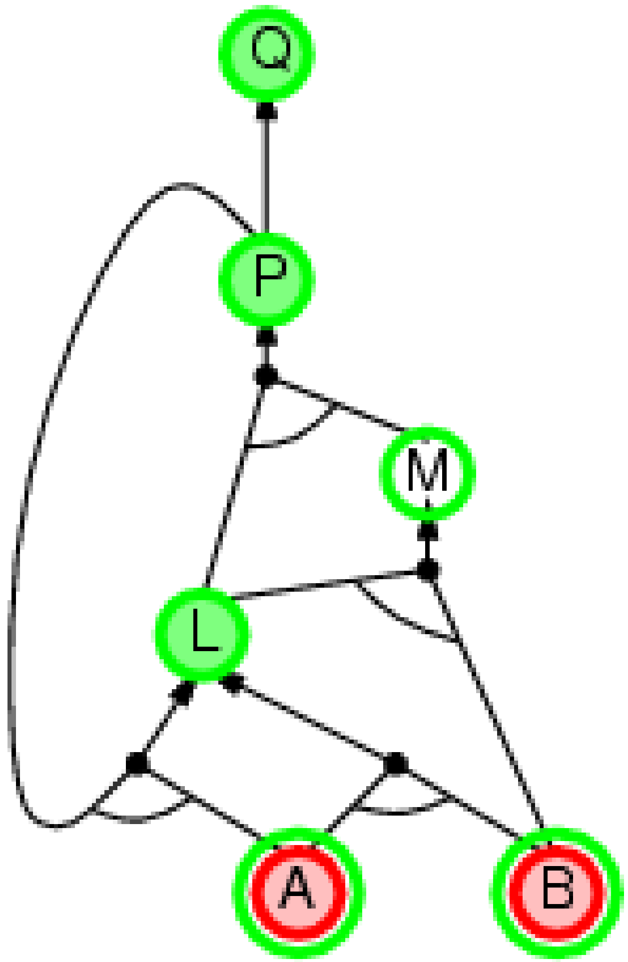


Backward chaining example



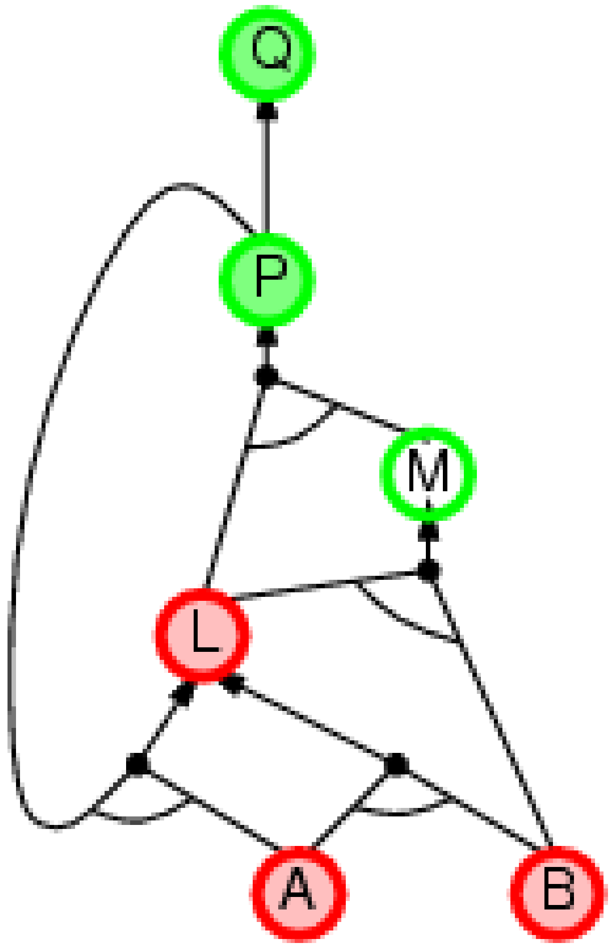
we need P to prove L and L to prove P.

Backward chaining example

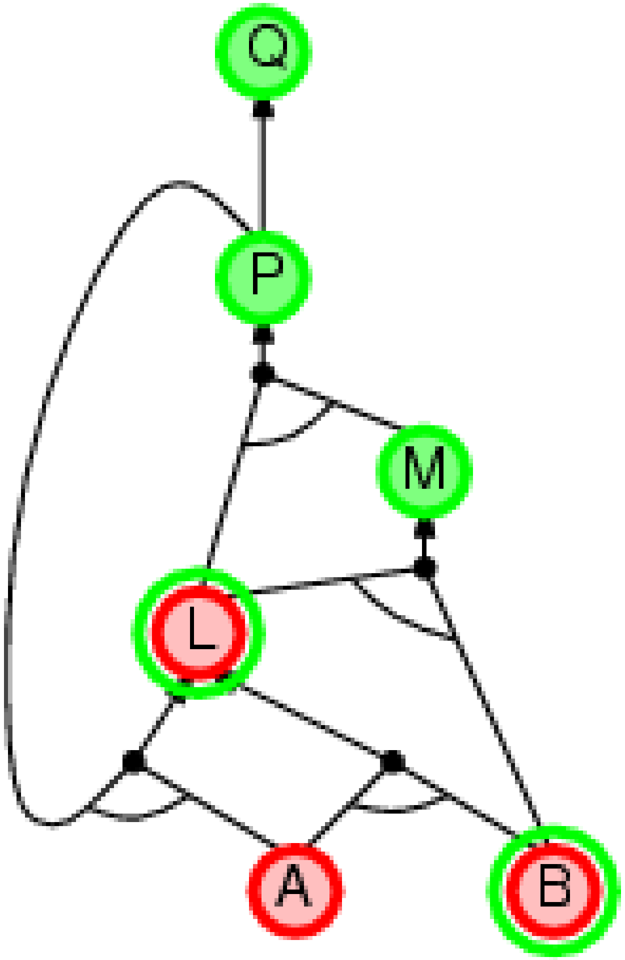


As soon as you can move forward, do so.

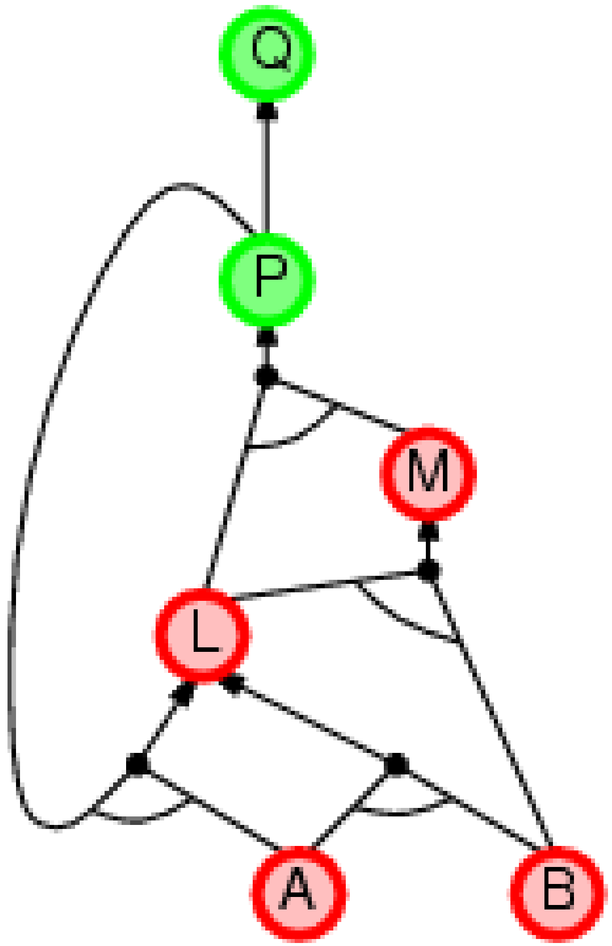
Backward chaining example



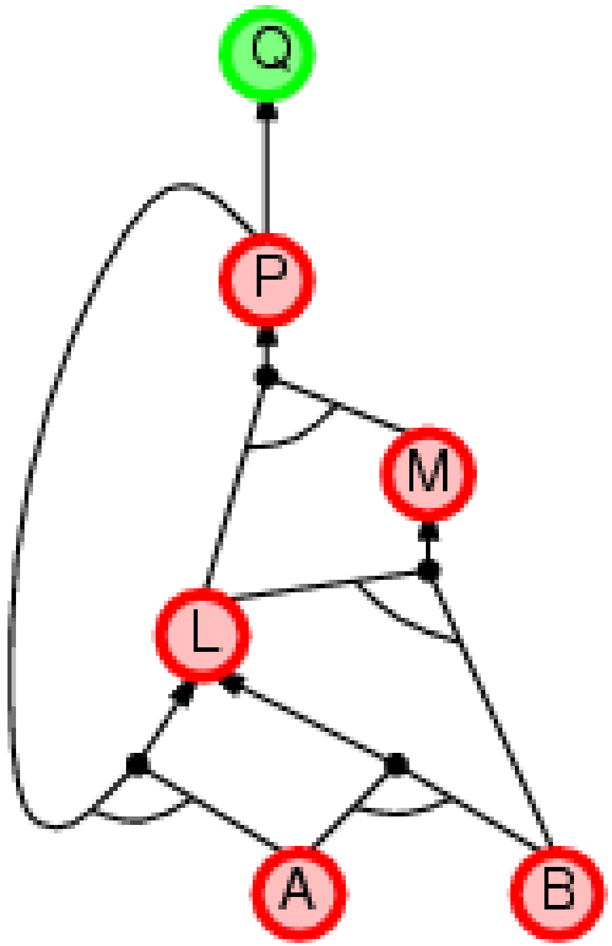
Backward chaining example



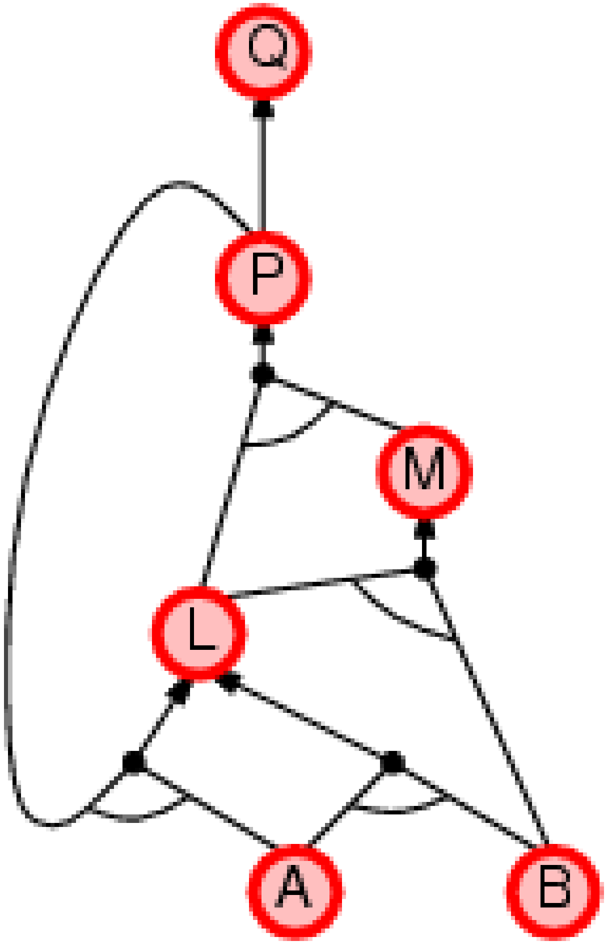
Backward chaining example



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

Model Checking

Two families of efficient algorithms:

- Complete backtracking search algorithms:
  - E.g., DPLL algorithm
- Incomplete local search algorithms
  - E.g., `WalkSAT` algorithm

The DPLL algorithm

Determine if an input propositional logic sentence (in CNF) is satisfiable. This is just backtracking search for a CSP.

Improvements:

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. (if there is a model for S, then making a pure symbol true is also a model).

3 Unit clause heuristic

   Unit clause: only one literal in the clause

   The only literal in a unit clause must be true.

   Note: literals can become a pure symbol or a unit clause when other literals obtain truth values. e.g.

$$(A \vee True) \wedge (\neg A \vee B)$$
$$A = pure$$

The `WalkSAT` algorithm

- Incomplete, local search algorithm
- Evaluation function: The min-conflict heuristic of minimizing the number of unsatisfied clauses
- Balance between greediness and randomness

Walksat Procedure

Start with random initial assignment.
Pick a random unsatisfied clause.
Select and flip a variable from that clause:
  With probability p, pick a random variable.
  With probability 1-p, pick greedily
    a variable that minimizes the number of unsatisfied clauses
Repeat to predefined maximum number flips;
  if no solution found, restart.

Hard satisfiability problems

- Consider *random* 3-CNF sentences. e.g.,

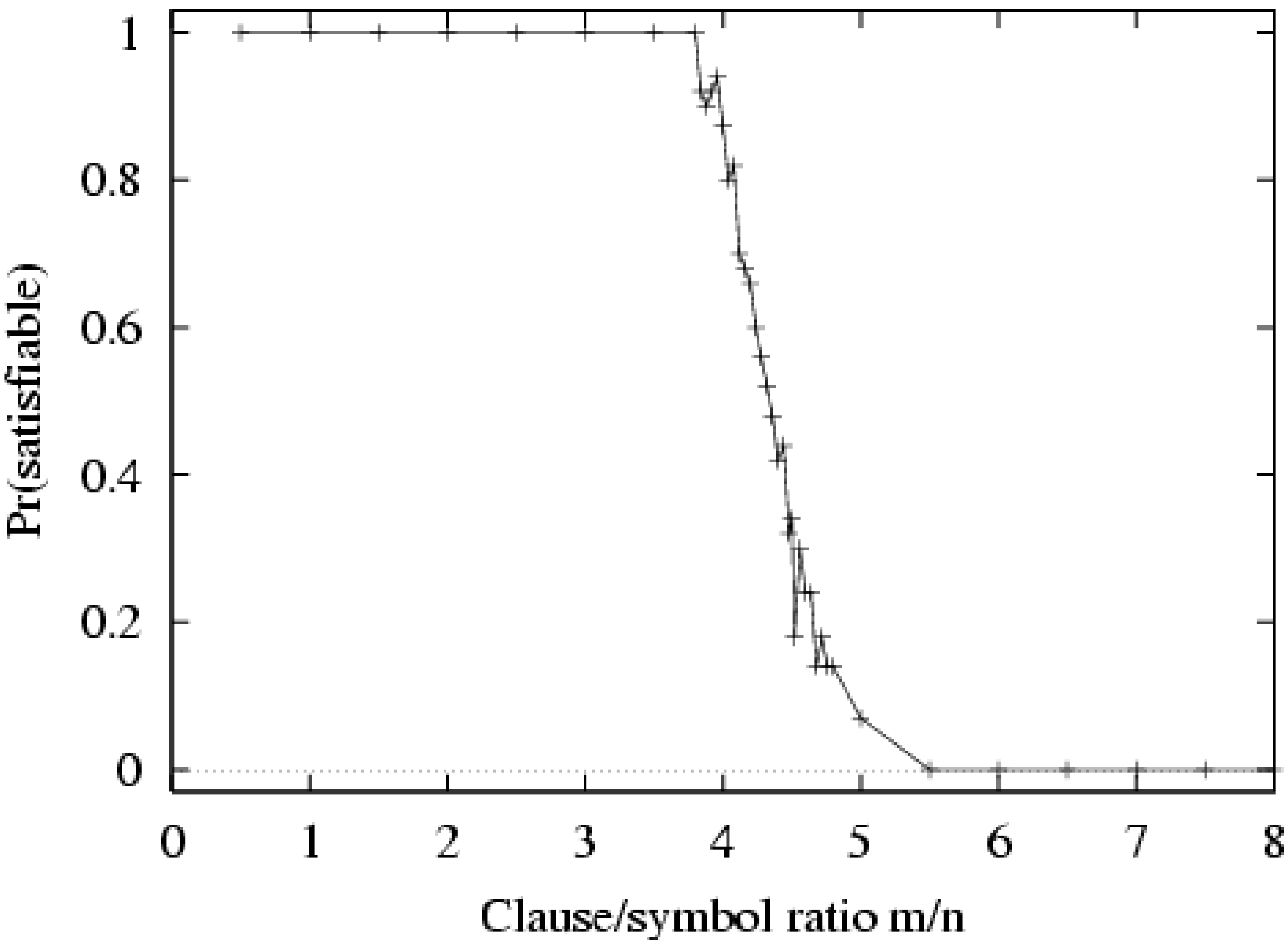
  $(\neg D \lor \neg B \lor C) \land (B \lor \neg A \lor \neg C) \land (\neg C \lor \neg B \lor E) \land (E \lor \neg D \lor B) \land (B \lor E \lor \neg C)$

  m = number of clauses (5)

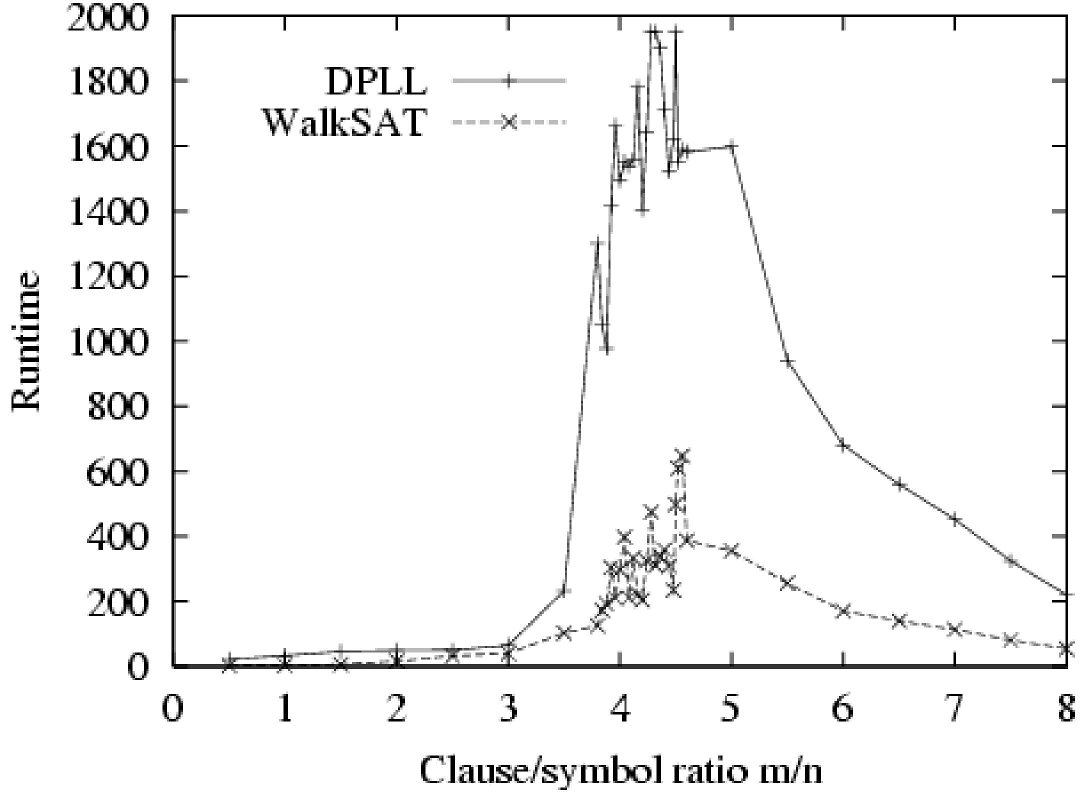
  n = number of symbols (5)

  – Hard problems seem to cluster near *m/n* = 4.3 (critical point)

Hard satisfiability problems



Hard satisfiability problems



- Median runtime for 100 satisfiable random 3-CNF sentences, $n = 50$

Common Sense Reasoning

Example, adapted from Lenat

You are told: John drove to the grocery store and bought a pound of noodles, a pound of ground beef, and two pounds of tomatoes.

- Is John 3 years old?
- Is John a child?
- What will John do with the purchases?
- Did John have any money?
- Does John have less money after going to the store?
- Did John buy at least two tomatoes?
- Were the tomatoes made in the supermarket?
- Did John buy any meat?
- Is John a vegetarian?
- Will the tomatoes fit in John's car?

- Can Propositional Logic support these inferences?

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 and backward chaining are linear-time, complete for Horn clauses

- Propositional logic lacks expressive power